

libstdc++

Generated by Doxygen 1.13.2

---

<b>1 Deprecated List</b>	<b>1</b>
<b>2 Todo List</b>	<b>2</b>
<b>3 Topic Documentation</b>	<b>4</b>
3.1 Algorithms . . . . .	4
3.1.1 Detailed Description . . . . .	5
3.1.2 Generalized Numeric operations . . . . .	5
3.1.3 Mutating . . . . .	19
3.1.4 Non-Mutating . . . . .	42
3.1.5 Sorting . . . . .	60
3.2 Atomics . . . . .	99
3.2.1 Detailed Description . . . . .	103
3.2.2 Macro Definition Documentation . . . . .	103
3.2.3 Typedef Documentation . . . . .	104
3.2.4 Enumeration Type Documentation . . . . .	110
3.2.5 Function Documentation . . . . .	110
3.3 Concurrency . . . . .	111
3.3.1 Detailed Description . . . . .	111
3.3.2 Condition Variables . . . . .	111
3.3.3 Futures . . . . .	112
3.3.4 Mutexes . . . . .	116
3.3.5 Threads . . . . .	119
3.4 Containers . . . . .	120
3.4.1 Detailed Description . . . . .	120
3.4.2 Associative . . . . .	121
3.4.3 Sequences . . . . .	121
3.4.4 Unordered Associative . . . . .	122
3.5 Diagnostics . . . . .	123
3.5.1 Detailed Description . . . . .	124
3.5.2 Function Documentation . . . . .	124
3.5.3 Exceptions . . . . .	127
3.6 Extensions . . . . .	133
3.6.1 Detailed Description . . . . .	134
3.6.2 Dynamic Bitset. . . . .	134
3.6.3 Policy-Based Data Structures . . . . .	138
3.6.4 SGI . . . . .	147
3.7 Filesystem . . . . .	157
3.7.1 Detailed Description . . . . .	161
3.7.2 Typedef Documentation . . . . .	161

3.7.3 Enumeration Type Documentation	161
3.7.4 Function Documentation	162
3.8 I/O	163
3.8.1 Detailed Description	164
3.8.2 Typedef Documentation	165
3.9 Iterators	167
3.9.1 Detailed Description	170
3.9.2 Function Documentation	170
3.9.3 Iterator Tags	173
3.10 Locales	173
3.10.1 Detailed Description	174
3.10.2 Function Documentation	174
3.11 Numerics	176
3.11.1 Detailed Description	176
3.11.2 Bit manipulation	177
3.11.3 Complex Numbers	177
3.11.4 Decimal Floating-Point Arithmetic	187
3.11.5 Mathematical Special Functions	188
3.11.6 Numeric Arrays	216
3.11.7 Random Number Generation	233
3.11.8 TR1 Mathematical Special Functions	246
3.12 Regular Expressions	252
3.12.1 Detailed Description	255
3.12.2 Typedef Documentation	255
3.12.3 Function Documentation	256
3.12.4 Base and Implementation Classes	272
3.13 Strings	273
3.13.1 Detailed Description	273
3.13.2 Typedef Documentation	273
3.14 Technical Specifications	274
3.14.1 Detailed Description	274
3.14.2 Filesystem TS	274
3.14.3 Library Fundamentals TS	280
3.14.4 Parallelism TS	294
3.15 Utilities	295
3.15.1 Detailed Description	298
3.15.2 Function Documentation	298
3.15.3 Variable Documentation	306
3.15.4 Function Objects	307

3.15.5 Memory	317
3.15.6 Metaprogramming	346
3.15.7 Rational Arithmetic	354
3.15.8 Time	356
<b>4 Namespace Documentation</b>	<b>377</b>
4.1 <code>__gnu_cxx</code> Namespace Reference	377
4.1.1 Detailed Description	394
4.1.2 Typedef Documentation	394
4.1.3 Function Documentation	394
4.2 <code>__gnu_cxx::__detail</code> Namespace Reference	404
4.2.1 Detailed Description	404
4.2.2 Function Documentation	404
4.3 <code>__gnu_cxx::typelist</code> Namespace Reference	405
4.3.1 Detailed Description	405
4.3.2 Function Documentation	405
4.4 <code>__gnu_debug</code> Namespace Reference	405
4.4.1 Detailed Description	411
4.4.2 Typedef Documentation	411
4.4.3 Enumeration Type Documentation	412
4.4.4 Function Documentation	412
4.5 <code>__gnu_internal</code> Namespace Reference	414
4.5.1 Detailed Description	414
4.6 <code>__gnu_parallel</code> Namespace Reference	414
4.6.1 Detailed Description	422
4.6.2 Typedef Documentation	422
4.6.3 Enumeration Type Documentation	423
4.6.4 Function Documentation	423
4.6.5 Variable Documentation	462
4.7 <code>__gnu_pbds</code> Namespace Reference	462
4.7.1 Detailed Description	464
4.8 <code>__gnu_sequential</code> Namespace Reference	464
4.8.1 Detailed Description	464
4.9 <code>abi</code> Namespace Reference	464
4.9.1 Detailed Description	464
4.10 <code>std</code> Namespace Reference	464
4.10.1 Detailed Description	578
4.10.2 Typedef Documentation	578
4.10.3 Enumeration Type Documentation	580



4.10.4 Function Documentation . . . . .	581
4.10.5 Variable Documentation . . . . .	646
4.11 std::__debug Namespace Reference . . . . .	647
4.11.1 Detailed Description . . . . .	653
4.11.2 Function Documentation . . . . .	654
4.12 std::__detail Namespace Reference . . . . .	654
4.12.1 Detailed Description . . . . .	656
4.12.2 Function Documentation . . . . .	656
4.13 std::__parallel Namespace Reference . . . . .	657
4.13.1 Detailed Description . . . . .	673
4.13.2 Function Documentation . . . . .	674
4.14 std::chrono Namespace Reference . . . . .	674
4.14.1 Detailed Description . . . . .	681
4.15 std::decimal Namespace Reference . . . . .	681
4.15.1 Detailed Description . . . . .	690
4.15.2 Function Documentation . . . . .	690
4.16 std::experimental Namespace Reference . . . . .	691
4.16.1 Detailed Description . . . . .	701
4.16.2 Function Documentation . . . . .	701
4.16.3 Variable Documentation . . . . .	703
4.17 std::filesystem Namespace Reference . . . . .	703
4.17.1 Detailed Description . . . . .	706
4.18 std::literals Namespace Reference . . . . .	706
4.18.1 Detailed Description . . . . .	707
4.19 std::literals::chrono_literals Namespace Reference . . . . .	707
4.19.1 Detailed Description . . . . .	708
4.20 std::placeholders Namespace Reference . . . . .	708
4.20.1 Detailed Description . . . . .	709
4.21 std::regex_constants Namespace Reference . . . . .	709
4.21.1 Detailed Description . . . . .	710
4.21.2 Enumeration Type Documentation . . . . .	710
4.21.3 Function Documentation . . . . .	711
4.21.4 Variable Documentation . . . . .	715
4.22 std::rel_ops Namespace Reference . . . . .	718
4.22.1 Detailed Description . . . . .	719
4.22.2 Function Documentation . . . . .	719
4.23 std::this_thread Namespace Reference . . . . .	720
4.23.1 Detailed Description . . . . .	720
4.23.2 Function Documentation . . . . .	721

4.24	<a href="#">std::tr1 Namespace Reference</a>	721
4.24.1	<a href="#">Detailed Description</a>	724
4.24.2	<a href="#">Function Documentation</a>	724
4.25	<a href="#">std::tr1::__detail Namespace Reference</a>	727
4.25.1	<a href="#">Detailed Description</a>	727
4.26	<a href="#">std::tr2 Namespace Reference</a>	727
4.26.1	<a href="#">Detailed Description</a>	728
4.27	<a href="#">std::tr2::__detail Namespace Reference</a>	728
4.27.1	<a href="#">Detailed Description</a>	728
<b>5</b>	<b>Class Documentation</b>	<b>728</b>
5.1	<a href="#">__gnu_parallel::__accumulate_binop_reduct&lt;_BinOp&gt; Struct Template Reference</a>	728
5.1.1	<a href="#">Detailed Description</a>	729
5.2	<a href="#">__gnu_parallel::__accumulate_selector&lt;_It&gt; Struct Template Reference</a>	729
5.2.1	<a href="#">Detailed Description</a>	729
5.2.2	<a href="#">Member Function Documentation</a>	730
5.2.3	<a href="#">Member Data Documentation</a>	730
5.3	<a href="#">__gnu_parallel::__adjacent_difference_selector&lt;_It&gt; Struct Template Reference</a>	730
5.3.1	<a href="#">Detailed Description</a>	731
5.3.2	<a href="#">Member Data Documentation</a>	731
5.4	<a href="#">__gnu_parallel::__adjacent_find_selector Struct Reference</a>	731
5.4.1	<a href="#">Detailed Description</a>	732
5.4.2	<a href="#">Member Function Documentation</a>	732
5.5	<a href="#">__gnu_cxx::__alloc_traits&lt;_Alloc, typename&gt; Struct Template Reference</a>	732
5.5.1	<a href="#">Detailed Description</a>	734
5.5.2	<a href="#">Member Typedef Documentation</a>	734
5.5.3	<a href="#">Member Function Documentation</a>	735
5.6	<a href="#">std::__basic_future&lt;_Res&gt; Class Template Reference</a>	738
5.6.1	<a href="#">Detailed Description</a>	739
5.6.2	<a href="#">Member Function Documentation</a>	740
5.7	<a href="#">__gnu_parallel::__binder1st&lt;_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType&gt; Class Template Reference</a>	740
5.7.1	<a href="#">Detailed Description</a>	741
5.7.2	<a href="#">Member Typedef Documentation</a>	741
5.8	<a href="#">__gnu_parallel::__binder2nd&lt;_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType&gt; Class Template Reference</a>	741
5.8.1	<a href="#">Detailed Description</a>	742
5.8.2	<a href="#">Member Typedef Documentation</a>	742
5.9	<a href="#">std::__codecvt_abstract_base&lt;_InternT, _ExternT, _StateT&gt; Class Template Reference</a>	742
5.9.1	<a href="#">Detailed Description</a>	743

5.9.2 Member Function Documentation	743
5.10 <code>__gnu_cxx::__common_pool_policy&lt; _PoolTp, _Thread &gt;</code> Struct Template Reference	746
5.10.1 Detailed Description	746
5.11 <code>__gnu_parallel::__count_if_selector&lt; _It, _Diff &gt;</code> Struct Template Reference	746
5.11.1 Detailed Description	747
5.11.2 Member Function Documentation	747
5.11.3 Member Data Documentation	747
5.12 <code>__gnu_parallel::__count_selector&lt; _It, _Diff &gt;</code> Struct Template Reference	747
5.12.1 Detailed Description	748
5.12.2 Member Function Documentation	748
5.12.3 Member Data Documentation	749
5.13 <code>std::__ctype_abstract_base&lt; _CharT &gt;</code> Class Template Reference	749
5.13.1 Detailed Description	751
5.13.2 Member Typedef Documentation	751
5.13.3 Member Function Documentation	751
5.14 <code>std::tr2::__dynamic_bitset_base&lt; _WordT, _Alloc &gt;</code> Struct Template Reference	762
5.14.1 Detailed Description	764
5.14.2 Member Data Documentation	764
5.15 <code>__gnu_parallel::__fill_selector&lt; _It &gt;</code> Struct Template Reference	764
5.15.1 Detailed Description	764
5.15.2 Member Function Documentation	765
5.15.3 Member Data Documentation	765
5.16 <code>__gnu_parallel::__find_first_of_selector&lt; _FIterator &gt;</code> Struct Template Reference	765
5.16.1 Detailed Description	766
5.16.2 Member Function Documentation	766
5.17 <code>__gnu_parallel::__find_if_selector</code> Struct Reference	767
5.17.1 Detailed Description	767
5.17.2 Member Function Documentation	767
5.18 <code>__gnu_parallel::__for_each_selector&lt; _It &gt;</code> Struct Template Reference	768
5.18.1 Detailed Description	769
5.18.2 Member Function Documentation	769
5.18.3 Member Data Documentation	769
5.19 <code>__cxxabiv1::__forced_unwind</code> Class Reference	769
5.19.1 Detailed Description	769
5.20 <code>__gnu_parallel::__generate_selector&lt; _It &gt;</code> Struct Template Reference	770
5.20.1 Detailed Description	770
5.20.2 Member Function Documentation	770
5.20.3 Member Data Documentation	771
5.21 <code>__gnu_parallel::__generic_find_selector</code> Struct Reference	771

5.21.1 Detailed Description	771
5.22 <code>__gnu_parallel::__generic_for_each_selector&lt;_It&gt;</code> Struct Template Reference	771
5.22.1 Detailed Description	773
5.22.2 Member Data Documentation	773
5.23 <code>__gnu_parallel::__identity_selector&lt;_It&gt;</code> Struct Template Reference	773
5.23.1 Detailed Description	773
5.23.2 Member Function Documentation	774
5.23.3 Member Data Documentation	774
5.24 <code>__gnu_parallel::__inner_product_selector&lt;_It, _It2, _Tp&gt;</code> Struct Template Reference	774
5.24.1 Detailed Description	775
5.24.2 Constructor & Destructor Documentation	775
5.24.3 Member Function Documentation	775
5.24.4 Member Data Documentation	776
5.25 <code>std::__is_fast_hash&lt;_Hash&gt;</code> Struct Template Reference	776
5.25.1 Detailed Description	776
5.26 <code>std::__is_location_invariant&lt;_Tp&gt;</code> Struct Template Reference	776
5.26.1 Detailed Description	776
5.27 <code>__gnu_parallel::__max_element_reduct&lt;_Compare, _It&gt;</code> Struct Template Reference	777
5.27.1 Detailed Description	777
5.28 <code>__gnu_parallel::__min_element_reduct&lt;_Compare, _It&gt;</code> Struct Template Reference	777
5.28.1 Detailed Description	777
5.29 <code>__gnu_cxx::__detail::__mini_vector&lt;_Tp&gt;</code> Class Template Reference	777
5.29.1 Detailed Description	778
5.30 <code>__gnu_parallel::__mismatch_selector</code> Struct Reference	778
5.30.1 Detailed Description	779
5.30.2 Member Function Documentation	779
5.31 <code>__gnu_cxx::__mt_alloc&lt;_Tp, _Poolp&gt;</code> Class Template Reference	780
5.31.1 Detailed Description	781
5.32 <code>__gnu_cxx::__mt_alloc_base&lt;_Tp&gt;</code> Class Template Reference	781
5.32.1 Detailed Description	782
5.33 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch&lt;__sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt;</code> Struct Template Reference	782
5.33.1 Detailed Description	782
5.34 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch&lt;true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt;</code> Struct Template Reference	782
5.34.1 Detailed Description	782
5.35 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch&lt;__sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt;</code> Struct Template Reference	783
5.35.1 Detailed Description	783

5.36	<a href="#">__gnu_parallel::__multiway_merge_4_variant_sentinel_switch&lt; true, _RAIterlterator, _RAIter3, __↵ DifferenceTp, _Compare &gt; Struct Template Reference</a>	. 783
5.36.1	Detailed Description	. 783
5.37	<a href="#">__gnu_parallel::__multiway_merge_k_variant_sentinel_switch&lt; __sentinels, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare &gt; Struct Template Reference</a>	. 783
5.37.1	Detailed Description	. 784
5.38	<a href="#">__gnu_parallel::__multiway_merge_k_variant_sentinel_switch&lt; false, __stable, _RAIterlterator, __↵ RAIter3, _DifferenceTp, _Compare &gt; Struct Template Reference</a>	. 784
5.38.1	Detailed Description	. 784
5.39	<a href="#">std::__new_allocator&lt; _Tp &gt; Class Template Reference</a>	. 784
5.39.1	Detailed Description	. 785
5.40	<a href="#">std::__numeric_limits_base Struct Reference</a>	. 786
5.40.1	Detailed Description	. 788
5.40.2	Member Data Documentation	. 788
5.41	<a href="#">__gnu_cxx::__per_type_pool_policy&lt; _Tp, _PoolTp, _Thread &gt; Struct Template Reference</a>	. 790
5.41.1	Detailed Description	. 791
5.42	<a href="#">__gnu_cxx::__pool&lt; _Thread &gt; Class Template Reference</a>	. 791
5.42.1	Detailed Description	. 791
5.43	<a href="#">__gnu_cxx::__pool&lt; false &gt; Class Reference</a>	. 791
5.43.1	Detailed Description	. 792
5.44	<a href="#">__gnu_cxx::__pool&lt; true &gt; Class Reference</a>	. 792
5.44.1	Detailed Description	. 793
5.45	<a href="#">__gnu_cxx::__pool_alloc&lt; _Tp &gt; Class Template Reference</a>	. 793
5.45.1	Detailed Description	. 794
5.46	<a href="#">__gnu_cxx::__pool_alloc_base Class Reference</a>	. 794
5.46.1	Detailed Description	. 795
5.47	<a href="#">__gnu_cxx::__pool_base Struct Reference</a>	. 795
5.47.1	Detailed Description	. 796
5.48	<a href="#">__gnu_cxx::__rc_string_base&lt; _CharT, _Traits, _Alloc &gt; Class Template Reference</a>	. 796
5.48.1	Detailed Description	. 799
5.49	<a href="#">std::tr2::__reflection_typelist&lt; _Elements &gt; Struct Template Reference</a>	. 799
5.49.1	Detailed Description	. 799
5.50	<a href="#">std::tr2::__reflection_typelist&lt; _First, _Rest... &gt; Struct Template Reference</a>	. 799
5.50.1	Detailed Description	. 799
5.51	<a href="#">std::tr2::__reflection_typelist&lt;&gt; Struct Reference</a>	. 800
5.51.1	Detailed Description	. 800
5.52	<a href="#">__gnu_parallel::__replace_if_selector&lt; _It, _Op, _Tp &gt; Struct Template Reference</a>	. 800
5.52.1	Detailed Description	. 800
5.52.2	Constructor & Destructor Documentation	. 801
5.52.3	Member Function Documentation	. 801

5.52.4 Member Data Documentation	801
5.53 <code>__gnu_parallel::__replace_selector&lt;_It, _Tp&gt;</code> Struct Template Reference	801
5.53.1 Detailed Description	802
5.53.2 Constructor & Destructor Documentation	802
5.53.3 Member Function Documentation	803
5.53.4 Member Data Documentation	803
5.54 <code>__gnu_cxx::__scoped_lock</code> Class Reference	803
5.54.1 Detailed Description	803
5.55 <code>__gnu_parallel::__transform1_selector&lt;_It&gt;</code> Struct Template Reference	804
5.55.1 Detailed Description	804
5.55.2 Member Function Documentation	804
5.55.3 Member Data Documentation	805
5.56 <code>__gnu_parallel::__transform2_selector&lt;_It&gt;</code> Struct Template Reference	805
5.56.1 Detailed Description	805
5.56.2 Member Function Documentation	805
5.56.3 Member Data Documentation	806
5.57 <code>__gnu_parallel::__unary_negate&lt;_Predicate, argument_type&gt;</code> Class Template Reference	806
5.57.1 Detailed Description	807
5.57.2 Member Typedef Documentation	807
5.58 <code>__gnu_cxx::__versa_string&lt;_CharT, _Traits, _Alloc, _Base&gt;</code> Class Template Reference	807
5.58.1 Detailed Description	810
5.58.2 Constructor & Destructor Documentation	811
5.58.3 Member Function Documentation	814
5.58.4 Member Data Documentation	858
5.59 <code>__gnu_debug::__After_nth_from&lt;_Iterator&gt;</code> Class Template Reference	858
5.59.1 Detailed Description	858
5.60 <code>std::_Base_bitset&lt;_Nw&gt;</code> Struct Template Reference	858
5.60.1 Detailed Description	859
5.60.2 Member Data Documentation	859
5.61 <code>std::_Base_bitset&lt;0&gt;</code> Struct Reference	859
5.61.1 Detailed Description	861
5.61.2 Member Data Documentation	861
5.62 <code>std::_Base_bitset&lt;1&gt;</code> Struct Reference	861
5.62.1 Detailed Description	862
5.62.2 Member Data Documentation	863
5.63 <code>__gnu_debug::__BeforeBeginHelper&lt;_Sequence&gt;</code> Struct Template Reference	863
5.63.1 Detailed Description	863
5.64 <code>std::_Bind&lt;_Signature&gt;</code> Class Template Reference	863
5.64.1 Detailed Description	863

5.65	<a href="#">std::__Bind_result&lt; _Result, _Signature &gt; Class Template Reference</a>	863
5.65.1	Detailed Description	863
5.66	<a href="#">__gnu_cxx::__detail::__Bitmap_counter&lt; _Tp &gt; Class Template Reference</a>	863
5.66.1	Detailed Description	864
5.67	<a href="#">std::__detail::__BracketMatcher&lt; _TraitsT, __icase, __collate &gt; Struct Template Reference</a>	864
5.67.1	Detailed Description	864
5.68	<a href="#">__gnu_cxx::__Caster&lt; _ToType &gt; Struct Template Reference</a>	865
5.68.1	Detailed Description	865
5.69	<a href="#">__gnu_cxx::__Char_types&lt; _CharT &gt; Struct Template Reference</a>	865
5.69.1	Detailed Description	865
5.70	<a href="#">__gnu_pbds::detail::pat_trie_base::__Clter&lt; Node, Leaf, Head, Inode, Is_Forward_Iterator &gt; Class Template Reference</a>	865
5.70.1	Detailed Description	867
5.71	<a href="#">std::__detail::__Compiler&lt; _TraitsT &gt; Class Template Reference</a>	867
5.71.1	Detailed Description	868
5.72	<a href="#">std::__parallel::__CRandNumber&lt; _MustBeInt &gt; Struct Template Reference</a>	868
5.72.1	Detailed Description	868
5.73	<a href="#">std::__Deque_base&lt; _Tp, _Alloc &gt; Class Template Reference</a>	868
5.73.1	Detailed Description	869
5.73.2	Member Function Documentation	869
5.74	<a href="#">std::__Deque_iterator&lt; _Tp, _Ref, _Ptr &gt; Struct Template Reference</a>	870
5.74.1	Detailed Description	871
5.74.2	Member Function Documentation	871
5.75	<a href="#">__gnu_parallel::__DRandomShufflingGlobalData&lt; _RAIter &gt; Struct Template Reference</a>	871
5.75.1	Detailed Description	872
5.75.2	Constructor & Destructor Documentation	872
5.75.3	Member Data Documentation	872
5.76	<a href="#">__gnu_parallel::__DRSSorterPU&lt; _RAIter, _RandomNumberGenerator &gt; Struct Template Reference</a>	873
5.76.1	Detailed Description	873
5.76.2	Member Data Documentation	873
5.77	<a href="#">__gnu_parallel::__DummyReduct Struct Reference</a>	874
5.77.1	Detailed Description	874
5.78	<a href="#">__gnu_debug::__Equal_to&lt; _Type &gt; Class Template Reference</a>	874
5.78.1	Detailed Description	875
5.79	<a href="#">__gnu_parallel::__EqualFromLess&lt; _T1, _T2, _Compare &gt; Class Template Reference</a>	875
5.79.1	Detailed Description	875
5.79.2	Member Typedef Documentation	876
5.80	<a href="#">__gnu_parallel::__EqualTo&lt; _T1, _T2 &gt; Struct Template Reference</a>	876
5.80.1	Detailed Description	877

5.80.2 Member Typedef Documentation	877
5.81 <code>std::__detail::__Executor&lt;_Biliter, _Alloc, _TraitsT, __dfs_mode&gt;</code> Class Template Reference	877
5.81.1 Detailed Description	878
5.82 <code>__gnu_cxx::ExtPtr_allocator&lt;_Tp&gt;</code> Class Template Reference	878
5.82.1 Detailed Description	879
5.83 <code>__gnu_cxx::__detail::__Ffit_finder&lt;_Tp&gt;</code> Class Template Reference	879
5.83.1 Detailed Description	879
5.84 <code>std::_Function_base</code> Class Reference	879
5.84.1 Detailed Description	880
5.85 <code>std::_Fwd_list_base&lt;_Tp, _Alloc&gt;</code> Struct Template Reference	880
5.85.1 Detailed Description	882
5.86 <code>std::_Fwd_list_const_iterator&lt;_Tp&gt;</code> Struct Template Reference	882
5.86.1 Detailed Description	883
5.86.2 Friends And Related Symbol Documentation	883
5.87 <code>std::_Fwd_list_iterator&lt;_Tp&gt;</code> Struct Template Reference	883
5.87.1 Detailed Description	884
5.87.2 Friends And Related Symbol Documentation	884
5.88 <code>std::_Fwd_list_node&lt;_Tp&gt;</code> Struct Template Reference	884
5.88.1 Detailed Description	885
5.89 <code>std::_Fwd_list_node_base</code> Struct Reference	885
5.89.1 Detailed Description	886
5.90 <code>__gnu_parallel::GuardedIterator&lt;_RAIter, _Compare&gt;</code> Class Template Reference	886
5.90.1 Detailed Description	886
5.90.2 Constructor & Destructor Documentation	886
5.90.3 Member Function Documentation	887
5.90.4 Friends And Related Symbol Documentation	887
5.91 <code>__gnu_pbds::detail::pat_trie_base::_Head&lt;_ATraits, Metadata&gt;</code> Struct Template Reference	888
5.91.1 Detailed Description	889
5.92 <code>__gnu_pbds::detail::pat_trie_base::_Inode&lt;_ATraits, Metadata&gt;</code> Struct Template Reference	889
5.92.1 Detailed Description	890
5.93 <code>__gnu_cxx::Invalid_type</code> Struct Reference	891
5.93.1 Detailed Description	891
5.94 <code>__gnu_pbds::detail::pat_trie_base::_Iter&lt;Node, Leaf, Head, Inode, Is_Forward_Iterator&gt;</code> Class Template Reference	891
5.94.1 Detailed Description	892
5.95 <code>std::_fwlist::_Iterator&lt;_Const, _Ptr&gt;</code> Class Template Reference	892
5.95.1 Detailed Description	893
5.95.2 Friends And Related Symbol Documentation	893
5.96 <code>__gnu_parallel::IteratorPair&lt;_Iterator1, _Iterator2, _IteratorCategory&gt;</code> Class Template Reference	893



5.96.1 Detailed Description	895
5.96.2 Member Typedef Documentation	895
5.96.3 Member Function Documentation	896
5.96.4 Friends And Related Symbol Documentation	896
5.96.5 Member Data Documentation	899
5.97 <code>__gnu_parallel::_IteratorTriple&lt;_Iterator1, _Iterator2, _Iterator3, _IteratorCategory&gt;</code> Class Template Reference	899
5.97.1 Detailed Description	900
5.98 <code>__gnu_parallel::_Job&lt;_DifferenceTp&gt;</code> Struct Template Reference	900
5.98.1 Detailed Description	900
5.98.2 Member Data Documentation	901
5.99 <code>__gnu_pbds::detail::pat_trie_base::_Leaf&lt;_ATraits, Metadata&gt;</code> Struct Template Reference	901
5.99.1 Detailed Description	902
5.100 <code>__gnu_parallel::_Less&lt;_T1, _T2&gt;</code> Struct Template Reference	902
5.100.1 Detailed Description	903
5.100.2 Member Typedef Documentation	903
5.101 <code>__gnu_parallel::_Lexicographic&lt;_T1, _T2, _Compare&gt;</code> Class Template Reference	903
5.101.1 Detailed Description	904
5.101.2 Member Typedef Documentation	904
5.102 <code>__gnu_parallel::_LexicographicReverse&lt;_T1, _T2, _Compare&gt;</code> Class Template Reference	905
5.102.1 Detailed Description	905
5.102.2 Member Typedef Documentation	905
5.103 <code>std::_List_base&lt;_Tp, _Alloc&gt;</code> Class Template Reference	906
5.103.1 Detailed Description	907
5.104 <code>std::_List_const_iterator&lt;_Tp&gt;</code> Struct Template Reference	907
5.104.1 Detailed Description	908
5.105 <code>std::_List_iterator&lt;_Tp&gt;</code> Struct Template Reference	908
5.105.1 Detailed Description	909
5.106 <code>std::_List_node&lt;_Tp&gt;</code> Struct Template Reference	909
5.106.1 Detailed Description	910
5.107 <code>std::_detail::_List_node_base</code> Struct Reference	910
5.107.1 Detailed Description	911
5.108 <code>std::_detail::_List_node_header</code> Struct Reference	911
5.108.1 Detailed Description	912
5.109 <code>__gnu_parallel::_LoserTreeBase&lt;_Tp, _Compare&gt;::_Loser</code> Struct Reference	912
5.109.1 Detailed Description	912
5.109.2 Member Data Documentation	912
5.110 <code>__gnu_parallel::_LoserTreePointerBase&lt;_Tp, _Compare&gt;::_Loser</code> Struct Reference	913
5.110.1 Detailed Description	913

5.111	<a href="#">__gnu_parallel::_LoserTree&lt; __stable, _Tp, _Compare &gt; Class Template Reference</a>	913
5.111.1	Detailed Description	914
5.111.2	Member Function Documentation	914
5.111.3	Member Data Documentation	915
5.112	<a href="#">__gnu_parallel::_LoserTree&lt; false, _Tp, _Compare &gt; Class Template Reference</a>	915
5.112.1	Detailed Description	916
5.112.2	Member Function Documentation	916
5.112.3	Member Data Documentation	917
5.113	<a href="#">__gnu_parallel::_LoserTreeBase&lt; _Tp, _Compare &gt; Class Template Reference</a>	917
5.113.1	Detailed Description	918
5.113.2	Constructor & Destructor Documentation	918
5.113.3	Member Function Documentation	919
5.113.4	Member Data Documentation	919
5.114	<a href="#">__gnu_parallel::_LoserTreePointer&lt; __stable, _Tp, _Compare &gt; Class Template Reference</a>	920
5.114.1	Detailed Description	920
5.115	<a href="#">__gnu_parallel::_LoserTreePointer&lt; false, _Tp, _Compare &gt; Class Template Reference</a>	921
5.115.1	Detailed Description	921
5.116	<a href="#">__gnu_parallel::_LoserTreePointerBase&lt; _Tp, _Compare &gt; Class Template Reference</a>	922
5.116.1	Detailed Description	922
5.117	<a href="#">__gnu_parallel::_LoserTreePointerUnguarded&lt; __stable, _Tp, _Compare &gt; Class Template Reference</a>	922
5.117.1	Detailed Description	923
5.118	<a href="#">__gnu_parallel::_LoserTreePointerUnguarded&lt; false, _Tp, _Compare &gt; Class Template Reference</a>	923
5.118.1	Detailed Description	924
5.119	<a href="#">__gnu_parallel::_LoserTreePointerUnguardedBase&lt; _Tp, _Compare &gt; Class Template Reference</a>	925
5.119.1	Detailed Description	925
5.120	<a href="#">__gnu_parallel::_LoserTreeTraits&lt; _Tp &gt; Struct Template Reference</a>	925
5.120.1	Detailed Description	926
5.120.2	Member Data Documentation	926
5.121	<a href="#">__gnu_parallel::_LoserTreeUnguarded&lt; __stable, _Tp, _Compare &gt; Class Template Reference</a>	926
5.121.1	Detailed Description	927
5.122	<a href="#">__gnu_parallel::_LoserTreeUnguarded&lt; false, _Tp, _Compare &gt; Class Template Reference</a>	927
5.122.1	Detailed Description	928
5.123	<a href="#">__gnu_parallel::_LoserTreeUnguardedBase&lt; _Tp, _Compare &gt; Class Template Reference</a>	928
5.123.1	Detailed Description	929
5.124	<a href="#">__gnu_pbds::detail::pat_trie_base::_Metadata&lt; Metadata, _Alloc &gt; Struct Template Reference</a>	929
5.124.1	Detailed Description	929
5.125	<a href="#">__gnu_pbds::detail::pat_trie_base::_Metadata&lt; null_type, _Alloc &gt; Struct Template Reference</a>	929
5.125.1	Detailed Description	930
5.126	<a href="#">__gnu_parallel::_Multiplies&lt; _Tp1, _Tp2, _Result &gt; Struct Template Reference</a>	930

5.126.1 Detailed Description	931
5.126.2 Member Typedef Documentation	931
5.127 std::__fwlist::__Node<_ValPtr> Struct Template Reference	931
5.127.1 Detailed Description	932
5.128 __gnu_pbds::detail::pat_trie_base::__Node_base<_ATraits, Metadata> Struct Template Reference	933
5.128.1 Detailed Description	933
5.129 std::__fwlist::__Node_base<_VoidPtr> Struct Template Reference	934
5.129.1 Detailed Description	934
5.130 __gnu_pbds::detail::pat_trie_base::__Node_citer<Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc> > Class Template Reference	934
5.130.1 Detailed Description	935
5.130.2 Member Typedef Documentation	935
5.130.3 Member Function Documentation	936
5.131 __gnu_pbds::detail::pat_trie_base::__Node_iter<Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc> Class Template Reference	937
5.131.1 Detailed Description	938
5.131.2 Member Typedef Documentation	938
5.131.3 Member Function Documentation	938
5.132 __gnu_debug::__Not_equal_to<_Type> Class Template Reference	940
5.132.1 Detailed Description	940
5.133 std::__Not_fn<_Fn> Class Template Reference	940
5.133.1 Detailed Description	941
5.134 __gnu_parallel::__Nothing Struct Reference	941
5.134.1 Detailed Description	941
5.134.2 Member Function Documentation	941
5.135 __gnu_parallel::__Piece<_DifferenceTp> Struct Template Reference	941
5.135.1 Detailed Description	942
5.135.2 Member Data Documentation	942
5.136 std::__Placeholder<_Num> Struct Template Reference	942
5.136.1 Detailed Description	942
5.137 __gnu_parallel::__Plus<_Tp1, _Tp2, _Result> Struct Template Reference	942
5.137.1 Detailed Description	943
5.137.2 Member Typedef Documentation	943
5.138 __gnu_parallel::__PMWMSSortingData<_RAIter> Struct Template Reference	944
5.138.1 Detailed Description	944
5.138.2 Member Data Documentation	944
5.139 __gnu_cxx::__Pointer_adapter<_Storage_policy> Class Template Reference	945
5.139.1 Detailed Description	947
5.140 __gnu_parallel::__PseudoSequence<_Tp, _DifferenceTp> Class Template Reference	947
5.140.1 Detailed Description	948

5.140.2 Constructor & Destructor Documentation	948
5.140.3 Member Function Documentation	948
5.141 <code>__gnu_parallel::PseudoSequenceIterator&lt; _Tp, _DifferenceTp &gt;</code> Class Template Reference	948
5.141.1 Detailed Description	949
5.142 <code>__gnu_parallel::QSBThreadLocal&lt; _RAIter &gt;</code> Struct Template Reference	949
5.142.1 Detailed Description	949
5.142.2 Member Typedef Documentation	950
5.142.3 Constructor & Destructor Documentation	950
5.142.4 Member Data Documentation	950
5.143 <code>std::__detail::__Quoted_string&lt; _String, _CharT &gt;</code> Struct Template Reference	951
5.143.1 Detailed Description	951
5.144 <code>__gnu_parallel::RandomNumber</code> Class Reference	951
5.144.1 Detailed Description	951
5.144.2 Constructor & Destructor Documentation	951
5.144.3 Member Function Documentation	952
5.145 <code>__gnu_cxx::Relative_pointer_impl&lt; _Tp &gt;</code> Class Template Reference	952
5.145.1 Detailed Description	953
5.146 <code>__gnu_cxx::Relative_pointer_impl&lt; const _Tp &gt;</code> Class Template Reference	954
5.146.1 Detailed Description	954
5.147 <code>__gnu_parallel::RestrictedBoundedConcurrentQueue&lt; _Tp &gt;</code> Class Template Reference	954
5.147.1 Detailed Description	955
5.147.2 Constructor & Destructor Documentation	955
5.147.3 Member Function Documentation	955
5.148 <code>__gnu_debug::Safe_container&lt; _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware &gt;</code> Class Template Reference	956
5.148.1 Detailed Description	958
5.149 <code>__gnu_debug::Safe_forward_list&lt; _SafeSequence &gt;</code> Class Template Reference	958
5.149.1 Detailed Description	958
5.149.2 Member Function Documentation	959
5.149.3 Member Data Documentation	959
5.150 <code>__gnu_debug::Safe_iterator&lt; _Iterator, _Sequence, _Category &gt;</code> Class Template Reference	960
5.150.1 Detailed Description	962
5.150.2 Constructor & Destructor Documentation	962
5.150.3 Member Function Documentation	963
5.150.4 Member Data Documentation	968
5.151 <code>__gnu_debug::Safe_iterator_base</code> Class Reference	969
5.151.1 Detailed Description	971
5.151.2 Constructor & Destructor Documentation	971
5.151.3 Member Function Documentation	972

5.151.4 Member Data Documentation	973
5.152 <code>__gnu_debug::_Safe_local_iterator&lt;_Iterator, _UContainer&gt;</code> Class Template Reference	974
5.152.1 Detailed Description	977
5.152.2 Constructor & Destructor Documentation	977
5.152.3 Member Function Documentation	978
5.152.4 Member Data Documentation	983
5.153 <code>__gnu_debug::_Safe_local_iterator_base</code> Class Reference	983
5.153.1 Detailed Description	985
5.153.2 Constructor & Destructor Documentation	985
5.153.3 Member Function Documentation	986
5.153.4 Member Data Documentation	988
5.154 <code>__gnu_debug::_Safe_node_sequence&lt;_Sequence&gt;</code> Class Template Reference	988
5.154.1 Detailed Description	989
5.154.2 Member Function Documentation	990
5.154.3 Member Data Documentation	991
5.155 <code>__gnu_debug::_Safe_sequence&lt;_Sequence&gt;</code> Class Template Reference	991
5.155.1 Detailed Description	992
5.155.2 Member Function Documentation	993
5.155.3 Member Data Documentation	994
5.156 <code>__gnu_debug::_Safe_sequence_base</code> Class Reference	994
5.156.1 Detailed Description	996
5.156.2 Constructor & Destructor Documentation	996
5.156.3 Member Function Documentation	996
5.156.4 Member Data Documentation	997
5.157 <code>__gnu_debug::_Safe_unordered_container&lt;_Container&gt;</code> Class Template Reference	997
5.157.1 Detailed Description	999
5.157.2 Member Function Documentation	999
5.157.3 Member Data Documentation	1000
5.158 <code>__gnu_debug::_Safe_unordered_container_base</code> Class Reference	1001
5.158.1 Detailed Description	1003
5.158.2 Constructor & Destructor Documentation	1003
5.158.3 Member Function Documentation	1003
5.158.4 Member Data Documentation	1004
5.159 <code>__gnu_debug::_Safe_vector&lt;_SafeSequence, _BaseSequence&gt;</code> Class Template Reference	1005
5.159.1 Detailed Description	1006
5.160 <code>__gnu_parallel::SamplingSorter&lt;__stable, _RAIter, _StrictWeakOrdering&gt;</code> Struct Template Reference	1006
5.160.1 Detailed Description	1006
5.161 <code>__gnu_parallel::SamplingSorter&lt;false, _RAIter, _StrictWeakOrdering&gt;</code> Struct Template Reference	1006
5.161.1 Detailed Description	1006

5.162	<a href="#">std::__detail::_Scanner&lt;_CharT&gt; Class Template Reference</a>	1007
5.162.1	Detailed Description	1008
5.162.2	Member Enumeration Documentation	1008
5.163	<a href="#">__gnu_debug::_Sequence_traits&lt;_Sequence&gt; Struct Template Reference</a>	1008
5.163.1	Detailed Description	1008
5.164	<a href="#">__gnu_parallel::_Settings Struct Reference</a>	1008
5.164.1	Detailed Description	1010
5.164.2	Member Function Documentation	1010
5.164.3	Member Data Documentation	1010
5.165	<a href="#">std::_Sp_ebo_helper&lt;_Nm, _Tp, false&gt; Struct Template Reference</a>	1015
5.165.1	Detailed Description	1015
5.166	<a href="#">std::_Sp_ebo_helper&lt;_Nm, _Tp, true&gt; Struct Template Reference</a>	1016
5.166.1	Detailed Description	1016
5.167	<a href="#">__gnu_parallel::_SplitConsistently&lt;__exact, _RAIter, _Compare, _SortingPlacesIterator&gt; Struct Template Reference</a>	1016
5.167.1	Detailed Description	1016
5.168	<a href="#">__gnu_parallel::_SplitConsistently&lt;false, _RAIter, _Compare, _SortingPlacesIterator&gt; Struct Template Reference</a>	1016
5.168.1	Detailed Description	1016
5.169	<a href="#">__gnu_parallel::_SplitConsistently&lt;true, _RAIter, _Compare, _SortingPlacesIterator&gt; Struct Template Reference</a>	1017
5.169.1	Detailed Description	1017
5.170	<a href="#">std::__detail::_StateSeq&lt;_TraitsT&gt; Class Template Reference</a>	1017
5.170.1	Detailed Description	1017
5.171	<a href="#">__gnu_cxx::_Std_pointer_impl&lt;_Tp&gt; Class Template Reference</a>	1017
5.171.1	Detailed Description	1018
5.172	<a href="#">__gnu_cxx::_Temporary_buffer&lt;_ForwardIterator, _Tp&gt; Class Template Reference</a>	1018
5.172.1	Detailed Description	1019
5.172.2	Constructor & Destructor Documentation	1019
5.172.3	Member Function Documentation	1019
5.173	<a href="#">std::_Temporary_buffer&lt;_ForwardIterator, _Tp&gt; Class Template Reference</a>	1020
5.173.1	Detailed Description	1020
5.173.2	Constructor & Destructor Documentation	1021
5.173.3	Member Function Documentation	1021
5.174	<a href="#">__gnu_cxx::_Unqualified_type&lt;_Tp&gt; Struct Template Reference</a>	1021
5.174.1	Detailed Description	1021
5.175	<a href="#">std::_Vector_base&lt;_Tp, _Alloc&gt; Struct Template Reference</a>	1022
5.175.1	Detailed Description	1023
5.176	<a href="#">std::add_const&lt;_Tp&gt; Struct Template Reference</a>	1023
5.176.1	Detailed Description	1023

5.177 <code>std::add_cv&lt; _Tp &gt;</code> Struct Template Reference . . . . .	1024
5.177.1 Detailed Description . . . . .	1024
5.178 <code>std::add_lvalue_reference&lt; _Tp &gt;</code> Struct Template Reference . . . . .	1024
5.178.1 Detailed Description . . . . .	1024
5.179 <code>std::add_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	1024
5.179.1 Detailed Description . . . . .	1024
5.180 <code>std::add_rvalue_reference&lt; _Tp &gt;</code> Struct Template Reference . . . . .	1024
5.180.1 Detailed Description . . . . .	1025
5.181 <code>std::add_volatile&lt; _Tp &gt;</code> Struct Template Reference . . . . .	1025
5.181.1 Detailed Description . . . . .	1025
5.182 <code>std::adopt_lock_t</code> Struct Reference . . . . .	1025
5.182.1 Detailed Description . . . . .	1025
5.183 <code>std::aligned_storage&lt; _Len, _Align &gt;</code> Struct Template Reference . . . . .	1025
5.183.1 Detailed Description . . . . .	1025
5.184 <code>std::aligned_union&lt; _Len, _Types &gt;</code> Struct Template Reference . . . . .	1026
5.184.1 Detailed Description . . . . .	1026
5.185 <code>std::alignment_of&lt; _Tp &gt;</code> Struct Template Reference . . . . .	1026
5.185.1 Detailed Description . . . . .	1027
5.186 <code>__gnu_cxx::allocator&lt; _Tp &gt;</code> Class Template Reference . . . . .	1027
5.186.1 Detailed Description . . . . .	1028
5.187 <code>std::allocator&lt; _Tp &gt;</code> Class Template Reference . . . . .	1029
5.187.1 Detailed Description . . . . .	1030
5.188 <code>std::allocator_traits&lt; _Alloc &gt;</code> Struct Template Reference . . . . .	1030
5.188.1 Detailed Description . . . . .	1032
5.188.2 Member Typedef Documentation . . . . .	1032
5.188.3 Member Function Documentation . . . . .	1034
5.189 <code>std::allocator_traits&lt; allocator&lt; _Tp &gt; &gt;</code> Struct Template Reference . . . . .	1036
5.189.1 Detailed Description . . . . .	1038
5.189.2 Member Typedef Documentation . . . . .	1038
5.189.3 Member Function Documentation . . . . .	1041
5.190 <code>std::allocator_traits&lt; allocator&lt; void &gt; &gt;</code> Struct Reference . . . . .	1045
5.190.1 Detailed Description . . . . .	1047
5.190.2 Member Typedef Documentation . . . . .	1047
5.190.3 Member Function Documentation . . . . .	1049
5.191 <code>std::allocator_traits&lt; pmr::polymorphic_allocator&lt; _Tp &gt; &gt;</code> Struct Template Reference . . . . .	1054
5.191.1 Detailed Description . . . . .	1055
5.191.2 Member Typedef Documentation . . . . .	1055
5.191.3 Member Function Documentation . . . . .	1058
5.192 <code>__gnu_cxx::limit_condition::always_adjustor</code> Struct Reference . . . . .	1062

5.192.1 Detailed Description	1063
5.193 <code>__gnu_cxx::random_condition::always_adjustor</code> Struct Reference	1063
5.193.1 Detailed Description	1063
5.194 <code>__gnu_cxx::annotate_base</code> Struct Reference	1063
5.194.1 Detailed Description	1064
5.195 <code>std::experimental::fundamentals_v1::any</code> Class Reference	1064
5.195.1 Detailed Description	1064
5.195.2 Constructor & Destructor Documentation	1065
5.195.3 Member Function Documentation	1065
5.196 <code>std::array&lt;_Tp, _Nm&gt;</code> Struct Template Reference	1066
5.196.1 Detailed Description	1067
5.197 <code>__gnu_pbds::associative_tag</code> Struct Reference	1068
5.197.1 Detailed Description	1068
5.198 <code>std::atomic&lt;_Tp&gt;</code> Struct Template Reference	1068
5.198.1 Detailed Description	1069
5.199 <code>std::atomic&lt;_Tp*&gt;</code> Struct Template Reference	1069
5.199.1 Detailed Description	1071
5.200 <code>std::atomic_flag</code> Struct Reference	1071
5.200.1 Detailed Description	1071
5.201 <code>std::auto_ptr&lt;_Tp&gt;</code> Class Template Reference	1071
5.201.1 Detailed Description	1072
5.201.2 Member Typedef Documentation	1072
5.201.3 Constructor & Destructor Documentation	1073
5.201.4 Member Function Documentation	1074
5.202 <code>std::auto_ptr_ref&lt;_Tp1&gt;</code> Struct Template Reference	1076
5.202.1 Detailed Description	1076
5.203 <code>std::back_insert_iterator&lt;_Container&gt;</code> Class Template Reference	1076
5.203.1 Detailed Description	1077
5.203.2 Member Typedef Documentation	1077
5.203.3 Constructor & Destructor Documentation	1078
5.203.4 Member Function Documentation	1078
5.204 <code>std::bad_alloc</code> Class Reference	1079
5.204.1 Detailed Description	1079
5.204.2 Member Function Documentation	1079
5.205 <code>std::experimental::fundamentals_v1::bad_any_cast</code> Class Reference	1079
5.205.1 Detailed Description	1080
5.205.2 Member Function Documentation	1080
5.206 <code>std::bad_cast</code> Class Reference	1080
5.206.1 Detailed Description	1081



5.206.2 Member Function Documentation	1081
5.207 std::bad_exception Class Reference	1081
5.207.1 Detailed Description	1082
5.207.2 Member Function Documentation	1082
5.208 std::bad_function_call Class Reference	1082
5.208.1 Detailed Description	1083
5.208.2 Member Function Documentation	1083
5.209 std::experimental::fundamentals_v1::bad_optional_access Class Reference	1083
5.209.1 Detailed Description	1083
5.209.2 Member Function Documentation	1084
5.210 std::bad_typeid Class Reference	1084
5.210.1 Detailed Description	1084
5.210.2 Member Function Documentation	1084
5.211 std::bad_weak_ptr Class Reference	1084
5.211.1 Detailed Description	1085
5.211.2 Member Function Documentation	1085
5.212 __gnu_parallel::balanced_quicksort_tag Struct Reference	1085
5.212.1 Detailed Description	1086
5.212.2 Member Function Documentation	1086
5.213 __gnu_parallel::balanced_tag Struct Reference	1087
5.213.1 Detailed Description	1087
5.213.2 Member Function Documentation	1087
5.214 std::tr2::bases< _Tp > Struct Template Reference	1088
5.214.1 Detailed Description	1088
5.215 __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc > Class Template Reference	1088
5.215.1 Detailed Description	1089
5.216 __gnu_pbds::basic_branch_tag Struct Reference	1089
5.216.1 Detailed Description	1089
5.217 std::basic_filebuf< _CharT, _Traits > Class Template Reference	1089
5.217.1 Detailed Description	1092
5.217.2 Constructor & Destructor Documentation	1093
5.217.3 Member Function Documentation	1093
5.217.4 Member Data Documentation	1108
5.218 std::basic_fstream< _CharT, _Traits > Class Template Reference	1110
5.218.1 Detailed Description	1117
5.218.2 Member Typedef Documentation	1117
5.218.3 Member Enumeration Documentation	1118
5.218.4 Constructor & Destructor Documentation	1118

5.218.5 Member Function Documentation	1120
5.218.6 Member Data Documentation	1155
5.219 <code>__gnu_pbds::basic_hash_table&lt; Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc &gt;</code> Class Template Reference	1160
5.219.1 Detailed Description	1161
5.220 <code>__gnu_pbds::basic_hash_tag</code> Struct Reference	1161
5.220.1 Detailed Description	1162
5.221 <code>std::basic_ifstream&lt; _CharT, _Traits &gt;</code> Class Template Reference	1162
5.221.1 Detailed Description	1168
5.221.2 Member Typedef Documentation	1168
5.221.3 Member Enumeration Documentation	1169
5.221.4 Constructor & Destructor Documentation	1169
5.221.5 Member Function Documentation	1171
5.221.6 Member Data Documentation	1198
5.222 <code>__gnu_pbds::basic_invalidation_guarantee</code> Struct Reference	1202
5.222.1 Detailed Description	1203
5.223 <code>std::basic_ios&lt; _CharT, _Traits &gt;</code> Class Template Reference	1203
5.223.1 Detailed Description	1206
5.223.2 Member Typedef Documentation	1207
5.223.3 Member Enumeration Documentation	1209
5.223.4 Constructor & Destructor Documentation	1209
5.223.5 Member Function Documentation	1210
5.223.6 Member Data Documentation	1221
5.224 <code>std::basic_iostream&lt; _CharT, _Traits &gt;</code> Class Template Reference	1225
5.224.1 Detailed Description	1231
5.224.2 Member Typedef Documentation	1231
5.224.3 Member Enumeration Documentation	1232
5.224.4 Constructor & Destructor Documentation	1232
5.224.5 Member Function Documentation	1233
5.224.6 Member Data Documentation	1267
5.225 <code>std::basic_istream&lt; _CharT, _Traits &gt;</code> Class Template Reference	1272
5.225.1 Detailed Description	1276
5.225.2 Member Typedef Documentation	1277
5.225.3 Member Enumeration Documentation	1278
5.225.4 Constructor & Destructor Documentation	1278
5.225.5 Member Function Documentation	1278
5.225.6 Member Data Documentation	1304
5.226 <code>std::basic_istream&lt; _CharT, _Traits, _Alloc &gt;</code> Class Template Reference	1309
5.226.1 Detailed Description	1314

5.226.2 Member Typedef Documentation . . . . .	1314
5.226.3 Member Enumeration Documentation . . . . .	1317
5.226.4 Constructor & Destructor Documentation . . . . .	1317
5.226.5 Member Function Documentation . . . . .	1318
5.226.6 Member Data Documentation . . . . .	1344
5.227 std::basic_ofstream< _CharT, _Traits > Class Template Reference . . . . .	1348
5.227.1 Detailed Description . . . . .	1353
5.227.2 Member Typedef Documentation . . . . .	1354
5.227.3 Member Enumeration Documentation . . . . .	1355
5.227.4 Constructor & Destructor Documentation . . . . .	1355
5.227.5 Member Function Documentation . . . . .	1356
5.227.6 Member Data Documentation . . . . .	1378
5.228 std::basic_ostream< _CharT, _Traits > Class Template Reference . . . . .	1382
5.228.1 Detailed Description . . . . .	1386
5.228.2 Member Typedef Documentation . . . . .	1387
5.228.3 Member Enumeration Documentation . . . . .	1388
5.228.4 Constructor & Destructor Documentation . . . . .	1388
5.228.5 Member Function Documentation . . . . .	1388
5.228.6 Member Data Documentation . . . . .	1408
5.229 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . . .	1412
5.229.1 Detailed Description . . . . .	1418
5.229.2 Member Typedef Documentation . . . . .	1419
5.229.3 Member Enumeration Documentation . . . . .	1420
5.229.4 Constructor & Destructor Documentation . . . . .	1420
5.229.5 Member Function Documentation . . . . .	1421
5.229.6 Member Data Documentation . . . . .	1442
5.230 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference . . . . .	1446
5.230.1 Detailed Description . . . . .	1447
5.230.2 Constructor & Destructor Documentation . . . . .	1448
5.230.3 Member Function Documentation . . . . .	1451
5.231 std::basic_streambuf< _CharT, _Traits > Class Template Reference . . . . .	1455
5.231.1 Detailed Description . . . . .	1458
5.231.2 Member Typedef Documentation . . . . .	1459
5.231.3 Constructor & Destructor Documentation . . . . .	1460
5.231.4 Member Function Documentation . . . . .	1460
5.231.5 Member Data Documentation . . . . .	1472
5.232 __gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference . . . . .	1473
5.232.1 Detailed Description . . . . .	1480
5.232.2 Member Function Documentation . . . . .	1481

5.232.3 Member Data Documentation	1545
5.233 <code>std::basic_string&lt;_CharT, _Traits, _Alloc&gt;</code> Class Template Reference	1545
5.233.1 Detailed Description	1553
5.233.2 Constructor & Destructor Documentation	1554
5.233.3 Member Function Documentation	1561
5.233.4 Member Data Documentation	1644
5.234 <code>std::basic_string_view&lt;_CharT, _Traits&gt;</code> Class Template Reference	1644
5.234.1 Detailed Description	1646
5.235 <code>std::experimental::filesystem::v1::basic_string_view&lt;_CharT, _Traits&gt;</code> Class Template Reference	1646
5.235.1 Detailed Description	1648
5.236 <code>std::experimental::fundamentals_v1::basic_string_view&lt;_CharT, _Traits&gt;</code> Class Template Reference	1648
5.236.1 Detailed Description	1650
5.237 <code>std::basic_stringbuf&lt;_CharT, _Traits, _Alloc&gt;</code> Class Template Reference	1650
5.237.1 Detailed Description	1653
5.237.2 Constructor & Destructor Documentation	1653
5.237.3 Member Function Documentation	1654
5.237.4 Member Data Documentation	1667
5.238 <code>std::basic_stringstream&lt;_CharT, _Traits, _Alloc&gt;</code> Class Template Reference	1668
5.238.1 Detailed Description	1675
5.238.2 Member Typedef Documentation	1675
5.238.3 Member Enumeration Documentation	1677
5.238.4 Constructor & Destructor Documentation	1677
5.238.5 Member Function Documentation	1678
5.238.6 Member Data Documentation	1712
5.239 <code>std::bernoulli_distribution</code> Class Reference	1717
5.239.1 Detailed Description	1717
5.239.2 Member Typedef Documentation	1717
5.239.3 Constructor & Destructor Documentation	1718
5.239.4 Member Function Documentation	1718
5.239.5 Friends And Related Symbol Documentation	1719
5.240 <code>std::bidirectional_iterator_tag</code> Struct Reference	1719
5.240.1 Detailed Description	1720
5.241 <code>__gnu_pbds::detail::bin_search_tree_const_it_&lt;Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc&gt;</code> Class Template Reference	1720
5.241.1 Detailed Description	1722
5.242 <code>__gnu_pbds::detail::bin_search_tree_const_node_it_&lt;Node, Const_Iterator, Iterator, _Alloc&gt;</code> Class Template Reference	1722
5.242.1 Detailed Description	1723
5.242.2 Member Typedef Documentation	1723
5.242.3 Member Function Documentation	1724

5.243	<a href="#">__gnu_pbds::detail::bin_search_tree_it_&lt; Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc &gt; Class Template Reference</a>	1725
5.243.1	Detailed Description	1727
5.244	<a href="#">__gnu_pbds::detail::bin_search_tree_node_it_&lt; Node, Const_Iterator, Iterator, _Alloc &gt; Class Template Reference</a>	1727
5.244.1	Detailed Description	1728
5.244.2	Member Typedef Documentation	1728
5.244.3	Member Function Documentation	1729
5.245	<a href="#">__gnu_pbds::detail::bin_search_tree_traits&lt; Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc &gt; Struct Template Reference</a>	1730
5.245.1	Detailed Description	1730
5.245.2	Member Typedef Documentation	1731
5.246	<a href="#">__gnu_pbds::detail::bin_search_tree_traits&lt; Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc &gt; Struct Template Reference</a>	1731
5.246.1	Detailed Description	1732
5.246.2	Member Typedef Documentation	1732
5.247	<a href="#">__gnu_cxx::binary_compose&lt; _Operation1, _Operation2, _Operation3 &gt; Class Template Reference</a>	1732
5.247.1	Detailed Description	1733
5.247.2	Member Typedef Documentation	1733
5.248	<a href="#">std::binary_function&lt; _Arg1, _Arg2, _Result &gt; Struct Template Reference</a>	1733
5.248.1	Detailed Description	1735
5.248.2	Member Typedef Documentation	1735
5.249	<a href="#">__gnu_pbds::detail::binary_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	1735
5.249.1	Detailed Description	1737
5.250	<a href="#">__gnu_pbds::detail::binary_heap_const_iterator_&lt; Value_Type, Entry, Simple, _Alloc &gt; Class Template Reference</a>	1737
5.250.1	Detailed Description	1738
5.250.2	Member Typedef Documentation	1738
5.250.3	Constructor & Destructor Documentation	1739
5.250.4	Member Function Documentation	1739
5.251	<a href="#">__gnu_pbds::detail::binary_heap_point_const_iterator_&lt; Value_Type, Entry, Simple, _Alloc &gt; Class Template Reference</a>	1740
5.251.1	Detailed Description	1742
5.251.2	Member Typedef Documentation	1742
5.251.3	Constructor & Destructor Documentation	1743
5.251.4	Member Function Documentation	1743
5.252	<a href="#">__gnu_pbds::binary_heap_tag Struct Reference</a>	1744
5.252.1	Detailed Description	1744
5.253	<a href="#">std::binary_negate&lt; _Predicate &gt; Class Template Reference</a>	1744
5.253.1	Detailed Description	1745
5.253.2	Member Typedef Documentation	1745

5.254 <a href="#">std::binder1st&lt; _Operation &gt; Class Template Reference</a>	1745
5.254.1 Detailed Description	1746
5.254.2 Member Typedef Documentation	1746
5.255 <a href="#">std::binder2nd&lt; _Operation &gt; Class Template Reference</a>	1747
5.255.1 Detailed Description	1747
5.255.2 Member Typedef Documentation	1748
5.256 <a href="#">std::binomial_distribution&lt; _IntType &gt; Class Template Reference</a>	1748
5.256.1 Detailed Description	1749
5.256.2 Member Typedef Documentation	1749
5.256.3 Member Function Documentation	1749
5.256.4 Friends And Related Symbol Documentation	1750
5.257 <a href="#">__gnu_pbds::detail::binomial_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	1752
5.257.1 Detailed Description	1754
5.258 <a href="#">__gnu_pbds::detail::binomial_heap_base&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	1754
5.258.1 Detailed Description	1756
5.259 <a href="#">__gnu_pbds::binomial_heap_tag Struct Reference</a>	1757
5.259.1 Detailed Description	1757
5.260 <a href="#">__gnu_cxx::bitmap_allocator&lt; _Tp &gt; Class Template Reference</a>	1757
5.260.1 Detailed Description	1758
5.260.2 Member Function Documentation	1758
5.261 <a href="#">std::__debug::bitset&lt; _Nb &gt; Class Template Reference</a>	1759
5.261.1 Detailed Description	1760
5.262 <a href="#">std::bitset&lt; _Nb &gt; Class Template Reference</a>	1760
5.262.1 Detailed Description	1762
5.262.2 Constructor & Destructor Documentation	1763
5.262.3 Member Function Documentation	1765
5.263 <a href="#">std::tr2::bool_set Class Reference</a>	1770
5.263.1 Detailed Description	1771
5.263.2 Constructor & Destructor Documentation	1771
5.263.3 Member Function Documentation	1771
5.264 <a href="#">__gnu_pbds::detail::branch_policy&lt; Node_Cltr, Node_Itr, _Alloc &gt; Struct Template Reference</a>	1772
5.264.1 Detailed Description	1773
5.265 <a href="#">__gnu_pbds::detail::branch_policy&lt; Node_Cltr, Node_Cltr, _Alloc &gt; Struct Template Reference</a>	1773
5.265.1 Detailed Description	1774
5.266 <a href="#">std::cauchy_distribution&lt; _RealType &gt; Class Template Reference</a>	1774
5.266.1 Detailed Description	1775
5.266.2 Member Typedef Documentation	1775
5.266.3 Member Function Documentation	1775
5.266.4 Friends And Related Symbol Documentation	1776

5.267	<a href="#">__gnu_pbds::cc_hash_max_collision_check_resize_trigger&lt; External_Load_Access, Size_Type &gt;</a>	
	Class Template Reference	1776
5.267.1	Detailed Description	1777
5.267.2	Member Enumeration Documentation	1777
5.267.3	Constructor & Destructor Documentation	1777
5.267.4	Member Function Documentation	1777
5.268	<a href="#">__gnu_pbds::cc_hash_table&lt; Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc &gt;</a>	
	Class Template Reference	1780
5.268.1	Detailed Description	1781
5.268.2	Constructor & Destructor Documentation	1781
5.269	<a href="#">__gnu_pbds::cc_hash_tag</a>	
	Struct Reference	1784
5.269.1	Detailed Description	1785
5.270	<a href="#">__gnu_pbds::detail::cc_ht_map&lt; Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy &gt;</a>	
	Class Template Reference	1785
5.270.1	Detailed Description	1788
5.270.2	Member Enumeration Documentation	1788
5.270.3	Member Function Documentation	1788
5.271	<a href="#">__gnu_cxx::char_traits&lt; _CharT &gt;</a>	
	Struct Template Reference	1790
5.271.1	Detailed Description	1791
5.272	<a href="#">std::char_traits&lt; _CharT &gt;</a>	
	Struct Template Reference	1791
5.272.1	Detailed Description	1792
5.273	<a href="#">std::char_traits&lt; __gnu_cxx::character&lt; _Value, _Int, _St &gt; &gt;</a>	
	Struct Template Reference	1792
5.273.1	Detailed Description	1794
5.274	<a href="#">std::char_traits&lt; wchar_t &gt;</a>	
	Struct Reference	1794
5.274.1	Detailed Description	1796
5.275	<a href="#">__gnu_cxx::character&lt; _Value, _Int, _St &gt;</a>	
	Struct Template Reference	1796
5.275.1	Detailed Description	1797
5.276	<a href="#">std::chi_squared_distribution&lt; _RealType &gt;</a>	
	Class Template Reference	1797
5.276.1	Detailed Description	1798
5.276.2	Member Typedef Documentation	1798
5.276.3	Member Function Documentation	1798
5.276.4	Friends And Related Symbol Documentation	1799
5.277	<a href="#">std::codecvt&lt; _InternT, _ExternT, _StateT &gt;</a>	
	Class Template Reference	1800
5.277.1	Detailed Description	1801
5.277.2	Member Function Documentation	1801
5.278	<a href="#">std::codecvt&lt; _InternT, _ExternT, encoding_state &gt;</a>	
	Class Template Reference	1805
5.278.1	Detailed Description	1807
5.278.2	Member Function Documentation	1807
5.279	<a href="#">std::codecvt&lt; char, char, mbstate_t &gt;</a>	
	Class Reference	1814
5.279.1	Detailed Description	1816

5.279.2 Member Function Documentation	1816
5.280 std::codecvt< char16_t, char, mbstate_t > Class Reference	1822
5.280.1 Detailed Description	1825
5.280.2 Member Function Documentation	1825
5.281 std::codecvt< char32_t, char, mbstate_t > Class Reference	1831
5.281.1 Detailed Description	1834
5.281.2 Member Function Documentation	1834
5.282 std::codecvt< wchar_t, char, mbstate_t > Class Reference	1840
5.282.1 Detailed Description	1843
5.282.2 Member Function Documentation	1843
5.283 std::codecvt_base Class Reference	1849
5.283.1 Detailed Description	1850
5.284 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference	1850
5.284.1 Detailed Description	1852
5.284.2 Member Function Documentation	1852
5.285 std::__cxx11::collate< _CharT > Class Template Reference	1856
5.285.1 Detailed Description	1857
5.285.2 Member Typedef Documentation	1857
5.285.3 Constructor & Destructor Documentation	1857
5.285.4 Member Function Documentation	1858
5.285.5 Member Data Documentation	1861
5.286 std::__cxx11::collate_byname< _CharT > Class Template Reference	1861
5.286.1 Detailed Description	1861
5.286.2 Member Typedef Documentation	1862
5.287 std::common_iterator< _It, _Sent > Class Template Reference	1862
5.287.1 Detailed Description	1863
5.288 std::common_type< _Tp > Struct Template Reference	1863
5.288.1 Detailed Description	1863
5.289 std::common_type< chrono::duration< _Rep, _Period > > Struct Template Reference	1863
5.289.1 Detailed Description	1863
5.290 std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > > Struct Template Reference	1864
5.290.1 Detailed Description	1864
5.291 std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > > Struct Template Reference	1864
5.291.1 Detailed Description	1864
5.292 std::common_type< chrono::time_point< _Clock, _Duration > > Struct Template Reference	1864
5.292.1 Detailed Description	1864
5.293 std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > > Struct Template Reference	1864



5.293.1 Detailed Description	1865
5.294 <code>std::common_type&lt; chrono::time_point&lt; _Clock, _Duration1 &gt;, chrono::time_point&lt; _Clock, _Duration2 &gt; &gt;</code> Struct Template Reference	1865
5.294.1 Detailed Description	1865
5.295 <code>std::compare_three_way_result&lt; _Tp, _Up &gt;</code> Struct Template Reference	1865
5.295.1 Detailed Description	1865
5.296 <code>std::complex&lt; _Tp &gt;</code> Class Template Reference	1865
5.296.1 Detailed Description	1866
5.296.2 Member Typedef Documentation	1866
5.296.3 Constructor & Destructor Documentation	1867
5.296.4 Member Function Documentation	1867
5.297 <code>std::complex&lt; double &gt;</code> Class Reference	1867
5.297.1 Detailed Description	1868
5.297.2 Member Typedef Documentation	1868
5.297.3 Constructor & Destructor Documentation	1869
5.297.4 Member Function Documentation	1869
5.298 <code>std::complex&lt; float &gt;</code> Class Reference	1870
5.298.1 Detailed Description	1871
5.298.2 Member Typedef Documentation	1871
5.298.3 Constructor & Destructor Documentation	1871
5.298.4 Member Function Documentation	1872
5.299 <code>std::complex&lt; long double &gt;</code> Class Reference	1873
5.299.1 Detailed Description	1874
5.299.2 Member Typedef Documentation	1874
5.299.3 Constructor & Destructor Documentation	1874
5.299.4 Member Function Documentation	1874
5.300 <code>__gnu_pbds::detail::cond_dealtor&lt; Entry, _Alloc &gt;</code> Class Template Reference	1876
5.300.1 Detailed Description	1876
5.301 <code>__gnu_pbds::detail::ov_tree_map&lt; Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc &gt;::cond_dtor&lt; Size_Type &gt;</code> Class Template Reference	1876
5.301.1 Detailed Description	1877
5.302 <code>__gnu_cxx::condition_base</code> Struct Reference	1877
5.302.1 Detailed Description	1877
5.303 <code>std::condition_variable</code> Class Reference	1877
5.303.1 Detailed Description	1878
5.304 <code>std::condition_variable_any</code> Class Reference	1878
5.304.1 Detailed Description	1879
5.305 <code>std::conditional&lt; _Cond, _Iftrue, _Iffalse &gt;</code> Struct Template Reference	1879
5.305.1 Detailed Description	1879
5.306 <code>__gnu_pbds::detail::pat_trie_base::_Inode&lt; _ATraits, Metadata &gt;::const_iterator</code> Struct Reference	1879

5.306.1 Detailed Description	1880
5.307 std::chrono::tzdb_list::const_iterator Class Reference	1880
5.307.1 Detailed Description	1881
5.308 std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	1881
5.308.1 Detailed Description	1881
5.308.2 Member Typedef Documentation	1882
5.309 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	1882
5.309.1 Detailed Description	1883
5.309.2 Member Typedef Documentation	1883
5.310 std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference	1883
5.310.1 Detailed Description	1884
5.310.2 Member Typedef Documentation	1884
5.311 std::const_mem_fun_t< _Ret, _Tp > Class Template Reference	1884
5.311.1 Detailed Description	1885
5.311.2 Member Typedef Documentation	1885
5.312 __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 > Struct Template Reference	1885
5.312.1 Detailed Description	1885
5.313 __gnu_parallel::constant_size_blocks_tag Struct Reference	1886
5.313.1 Detailed Description	1886
5.314 __gnu_cxx::constant_unary_fun< _Result, _Argument > Struct Template Reference	1886
5.314.1 Detailed Description	1886
5.315 __gnu_cxx::constant_void_fun< _Result > Struct Template Reference	1887
5.315.1 Detailed Description	1887
5.316 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI > Struct Template Reference	1887
5.316.1 Detailed Description	1887
5.317 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type > Struct Template Reference	1887
5.317.1 Detailed Description	1887
5.317.2 Member Typedef Documentation	1888
5.318 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type > Struct Template Reference	1888
5.318.1 Detailed Description	1888
5.318.2 Member Typedef Documentation	1888
5.319 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type > Struct Template Reference	1888
5.319.1 Detailed Description	1888
5.319.2 Member Typedef Documentation	1889
5.320 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type > Struct Template Reference	1889

5.320.1 Detailed Description	1889
5.320.2 Member Typedef Documentation	1889
5.321 <code>__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type &gt;</code> Struct Template Reference	1889
5.321.1 Detailed Description	1889
5.321.2 Member Typedef Documentation	1890
5.322 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, cc_hash_tag, Policy_TI &gt;</code> Struct Template Reference	1890
5.322.1 Detailed Description	1890
5.322.2 Member Typedef Documentation	1890
5.323 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, gp_hash_tag, Policy_TI &gt;</code> Struct Template Reference	1890
5.323.1 Detailed Description	1890
5.323.2 Member Typedef Documentation	1891
5.324 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, list_update_tag, Policy_TI &gt;</code> Struct Template Reference	1891
5.324.1 Detailed Description	1891
5.324.2 Member Typedef Documentation	1891
5.325 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, ov_tree_tag, Policy_TI &gt;</code> Struct Template Reference	1891
5.325.1 Detailed Description	1891
5.325.2 Member Typedef Documentation	1892
5.326 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, pat_trie_tag, Policy_TI &gt;</code> Struct Template Reference	1892
5.326.1 Detailed Description	1892
5.327 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, rb_tree_tag, Policy_TI &gt;</code> Struct Template Reference	1892
5.327.1 Detailed Description	1892
5.327.2 Member Typedef Documentation	1892
5.328 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, splay_tree_tag, Policy_TI &gt;</code> Struct Template Reference	1893
5.328.1 Detailed Description	1893
5.328.2 Member Typedef Documentation	1893
5.329 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, cc_hash_tag, Policy_TI &gt;</code> Struct Template Reference	1893
5.329.1 Detailed Description	1893
5.329.2 Member Typedef Documentation	1893
5.330 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, gp_hash_tag, Policy_TI &gt;</code> Struct Template Reference	1894
5.330.1 Detailed Description	1894
5.330.2 Member Typedef Documentation	1894

5.331	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, list_update_tag, Policy_Tl &gt; Struct Template Reference</a>	1894
5.331.1	<a href="#">Detailed Description</a>	1894
5.331.2	<a href="#">Member Typedef Documentation</a>	1894
5.332	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, ov_tree_tag, Policy_Tl &gt; Struct Template Reference</a>	1895
5.332.1	<a href="#">Detailed Description</a>	1895
5.332.2	<a href="#">Member Typedef Documentation</a>	1895
5.333	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, pat_trie_tag, Policy_Tl &gt; Struct Template Reference</a>	1895
5.333.1	<a href="#">Detailed Description</a>	1895
5.333.2	<a href="#">Member Typedef Documentation</a>	1895
5.334	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, rb_tree_tag, Policy_Tl &gt; Struct Template Reference</a>	1896
5.334.1	<a href="#">Detailed Description</a>	1896
5.335	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, splay_tree_tag, Policy_Tl &gt; Struct Template Reference</a>	1896
5.335.1	<a href="#">Detailed Description</a>	1896
5.335.2	<a href="#">Member Typedef Documentation</a>	1896
5.336	<a href="#">__gnu_pbds::container_error Struct Reference</a>	1896
5.336.1	<a href="#">Detailed Description</a>	1897
5.336.2	<a href="#">Member Function Documentation</a>	1897
5.337	<a href="#">__gnu_pbds::container_tag Struct Reference</a>	1897
5.337.1	<a href="#">Detailed Description</a>	1898
5.338	<a href="#">__gnu_pbds::container_traits&lt; Cntnr &gt; Struct Template Reference</a>	1898
5.338.1	<a href="#">Detailed Description</a>	1899
5.338.2	<a href="#">Member Enumeration Documentation</a>	1899
5.339	<a href="#">__gnu_pbds::container_traits_base&lt; _Tag &gt; Struct Template Reference</a>	1899
5.339.1	<a href="#">Detailed Description</a>	1899
5.340	<a href="#">__gnu_pbds::container_traits_base&lt; binary_heap_tag &gt; Struct Reference</a>	1899
5.340.1	<a href="#">Detailed Description</a>	1900
5.341	<a href="#">__gnu_pbds::container_traits_base&lt; binomial_heap_tag &gt; Struct Reference</a>	1900
5.341.1	<a href="#">Detailed Description</a>	1900
5.342	<a href="#">__gnu_pbds::container_traits_base&lt; cc_hash_tag &gt; Struct Reference</a>	1900
5.342.1	<a href="#">Detailed Description</a>	1900
5.343	<a href="#">__gnu_pbds::container_traits_base&lt; gp_hash_tag &gt; Struct Reference</a>	1900
5.343.1	<a href="#">Detailed Description</a>	1900
5.344	<a href="#">__gnu_pbds::container_traits_base&lt; list_update_tag &gt; Struct Reference</a>	1901
5.344.1	<a href="#">Detailed Description</a>	1901
5.345	<a href="#">__gnu_pbds::container_traits_base&lt; ov_tree_tag &gt; Struct Reference</a>	1901

5.345.1 Detailed Description	1901
5.346 <a href="#">__gnu_pbds::container_traits_base&lt; pairing_heap_tag &gt; Struct Reference</a>	1901
5.346.1 Detailed Description	1901
5.347 <a href="#">__gnu_pbds::container_traits_base&lt; pat_trie_tag &gt; Struct Reference</a>	1901
5.347.1 Detailed Description	1902
5.348 <a href="#">__gnu_pbds::container_traits_base&lt; rb_tree_tag &gt; Struct Reference</a>	1902
5.348.1 Detailed Description	1902
5.349 <a href="#">__gnu_pbds::container_traits_base&lt; rc_binomial_heap_tag &gt; Struct Reference</a>	1902
5.349.1 Detailed Description	1902
5.350 <a href="#">__gnu_pbds::container_traits_base&lt; splay_tree_tag &gt; Struct Reference</a>	1902
5.350.1 Detailed Description	1903
5.351 <a href="#">__gnu_pbds::container_traits_base&lt; thin_heap_tag &gt; Struct Reference</a>	1903
5.351.1 Detailed Description	1903
5.352 <a href="#">std::contiguous_iterator_tag Struct Reference</a>	1903
5.352.1 Detailed Description	1904
5.353 <a href="#">std::copyable_function&lt; _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)&gt; Class Template Reference</a>	1904
5.353.1 Detailed Description	1905
5.353.2 Constructor & Destructor Documentation	1906
5.353.3 Member Function Documentation	1907
5.353.4 Friends And Related Symbol Documentation	1908
5.354 <a href="#">std::counted_iterator&lt; _It &gt; Class Template Reference</a>	1909
5.354.1 Detailed Description	1910
5.355 <a href="#">std::ctype&lt; _CharT &gt; Class Template Reference</a>	1910
5.355.1 Detailed Description	1912
5.355.2 Member Function Documentation	1912
5.355.3 Member Data Documentation	1923
5.356 <a href="#">std::ctype&lt; char &gt; Class Reference</a>	1923
5.356.1 Detailed Description	1926
5.356.2 Member Typedef Documentation	1926
5.356.3 Constructor & Destructor Documentation	1926
5.356.4 Member Function Documentation	1927
5.356.5 Member Data Documentation	1958
5.357 <a href="#">std::ctype&lt; wchar_t &gt; Class Reference</a>	1958
5.357.1 Detailed Description	1960
5.357.2 Member Typedef Documentation	1960
5.357.3 Constructor & Destructor Documentation	1960
5.357.4 Member Function Documentation	1961
5.357.5 Member Data Documentation	1976

5.358 <code>std::ctype_base</code> Struct Reference	1976
5.358.1 Detailed Description	1977
5.359 <code>std::ctype_byname&lt;_CharT&gt;</code> Class Template Reference	1977
5.359.1 Detailed Description	1979
5.359.2 Member Function Documentation	1980
5.359.3 Member Data Documentation	1990
5.360 <code>std::ctype_byname&lt;char&gt;</code> Class Reference	1990
5.360.1 Detailed Description	1993
5.360.2 Member Function Documentation	1993
5.360.3 Member Data Documentation	2013
5.361 <code>__gnu_cxx::debug_allocator&lt;_Alloc&gt;</code> Class Template Reference	2013
5.361.1 Detailed Description	2014
5.362 <code>std::decay&lt;_Tp&gt;</code> Struct Template Reference	2014
5.362.1 Detailed Description	2014
5.363 <code>std::decimal::decimal128</code> Class Reference	2014
5.363.1 Detailed Description	2015
5.363.2 Constructor & Destructor Documentation	2015
5.364 <code>std::decimal::decimal32</code> Class Reference	2016
5.364.1 Detailed Description	2017
5.364.2 Constructor & Destructor Documentation	2017
5.365 <code>std::decimal::decimal64</code> Class Reference	2017
5.365.1 Detailed Description	2018
5.365.2 Constructor & Destructor Documentation	2018
5.366 <code>simd_abi::deduce&lt;_Tp, _Np, ...&gt;</code> Struct Template Reference	2019
5.366.1 Detailed Description	2019
5.367 <code>__gnu_pbds::detail::default_comb_hash_fn</code> Struct Reference	2019
5.367.1 Detailed Description	2019
5.367.2 Member Typedef Documentation	2019
5.368 <code>std::default_delete&lt;_Tp&gt;</code> Struct Template Reference	2019
5.368.1 Detailed Description	2020
5.368.2 Constructor & Destructor Documentation	2020
5.368.3 Member Function Documentation	2020
5.369 <code>std::default_delete&lt;_Tp[]&gt;</code> Struct Template Reference	2020
5.369.1 Detailed Description	2021
5.369.2 Constructor & Destructor Documentation	2021
5.369.3 Member Function Documentation	2021
5.370 <code>__gnu_pbds::detail::default_eq_fn&lt;Key&gt;</code> Struct Template Reference	2022
5.370.1 Detailed Description	2022
5.370.2 Member Typedef Documentation	2022

5.371	<a href="#">__gnu_pbds::detail::default_hash_fn&lt; Key &gt; Struct Template Reference</a>	2022
5.371.1	Detailed Description	2022
5.371.2	Member Typedef Documentation	2022
5.372	<a href="#">__gnu_parallel::default_parallel_tag Struct Reference</a>	2023
5.372.1	Detailed Description	2023
5.372.2	Member Function Documentation	2023
5.373	<a href="#">__gnu_pbds::detail::default_probe_fn&lt; Comb_Probe_Fn &gt; Struct Template Reference</a>	2024
5.373.1	Detailed Description	2024
5.373.2	Member Typedef Documentation	2024
5.374	<a href="#">__gnu_pbds::detail::default_resize_policy&lt; Comb_Hash_Fn &gt; Struct Template Reference</a>	2024
5.374.1	Detailed Description	2024
5.374.2	Member Typedef Documentation	2024
5.375	<a href="#">std::default_sentinel_t Struct Reference</a>	2024
5.375.1	Detailed Description	2025
5.376	<a href="#">__gnu_pbds::detail::default_trie_access_traits&lt; Key &gt; Struct Template Reference</a>	2025
5.376.1	Detailed Description	2025
5.377	<a href="#">__gnu_pbds::detail::default_trie_access_traits&lt; std::basic_string&lt; Char, Char_Traits, std::allocator&lt; char &gt; &gt; &gt; Struct Template Reference</a>	2025
5.377.1	Detailed Description	2025
5.377.2	Member Typedef Documentation	2025
5.378	<a href="#">__gnu_pbds::detail::default_update_policy Struct Reference</a>	2026
5.378.1	Detailed Description	2026
5.378.2	Member Typedef Documentation	2026
5.379	<a href="#">std::defer_lock_t Struct Reference</a>	2026
5.379.1	Detailed Description	2026
5.380	<a href="#">std::__debug::deque&lt; _Tp, _Allocator &gt; Class Template Reference</a>	2026
5.380.1	Detailed Description	2028
5.381	<a href="#">std::deque&lt; _Tp, _Alloc &gt; Class Template Reference</a>	2028
5.381.1	Detailed Description	2032
5.381.2	Constructor & Destructor Documentation	2033
5.381.3	Member Function Documentation	2036
5.382	<a href="#">std::destroying_delete_t Struct Reference</a>	2051
5.382.1	Detailed Description	2051
5.383	<a href="#">std::tr2::direct_bases&lt; _Tp &gt; Struct Template Reference</a>	2051
5.383.1	Detailed Description	2051
5.384	<a href="#">__gnu_pbds::direct_mask_range_hashing&lt; Size_Type &gt; Class Template Reference</a>	2051
5.384.1	Detailed Description	2052
5.384.2	Member Function Documentation	2052
5.385	<a href="#">__gnu_pbds::direct_mod_range_hashing&lt; Size_Type &gt; Class Template Reference</a>	2053

5.385.1 Detailed Description	2053
5.385.2 Member Function Documentation	2053
5.386 std::filesystem::directory_entry Class Reference	2054
5.386.1 Detailed Description	2055
5.387 std::filesystem::directory_iterator Class Reference	2055
5.387.1 Detailed Description	2056
5.388 std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference	2056
5.388.1 Detailed Description	2057
5.388.2 Member Typedef Documentation	2057
5.388.3 Constructor & Destructor Documentation	2057
5.388.4 Member Function Documentation	2058
5.388.5 Friends And Related Symbol Documentation	2060
5.389 std::discrete_distribution< _IntType > Class Template Reference	2061
5.389.1 Detailed Description	2062
5.389.2 Member Typedef Documentation	2062
5.389.3 Member Function Documentation	2062
5.389.4 Friends And Related Symbol Documentation	2063
5.390 std::divides< _Tp > Struct Template Reference	2064
5.390.1 Detailed Description	2065
5.390.2 Member Typedef Documentation	2065
5.391 std::divides< void > Struct Reference	2065
5.391.1 Detailed Description	2066
5.391.2 Member Typedef Documentation	2066
5.392 std::domain_error Class Reference	2066
5.392.1 Detailed Description	2067
5.392.2 Member Function Documentation	2067
5.393 __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc > Struct Template Reference	2067
5.393.1 Detailed Description	2068
5.394 std::chrono::duration< _Rep, _Period > Class Template Reference	2068
5.394.1 Detailed Description	2069
5.395 std::chrono::duration_values< _Rep > Struct Template Reference	2069
5.395.1 Detailed Description	2069
5.396 std::tr2::dynamic_bitset< _WordT, _Alloc > Class Template Reference	2069
5.396.1 Detailed Description	2072
5.396.2 Constructor & Destructor Documentation	2073
5.396.3 Member Function Documentation	2074
5.397 std::enable_if< bool, _Tp > Struct Template Reference	2083
5.397.1 Detailed Description	2083
5.398 std::enable_shared_from_this< _Tp > Class Template Reference	2083



5.398.1 Detailed Description	2083
5.399 <a href="#">__gnu_cxx::enc_filebuf&lt;_CharT&gt;</a> Class Template Reference	2083
5.399.1 Detailed Description	2086
5.399.2 Member Function Documentation	2086
5.399.3 Member Data Documentation	2099
5.400 <a href="#">__gnu_cxx::encoding_char_traits&lt;_CharT&gt;</a> Struct Template Reference	2102
5.400.1 Detailed Description	2103
5.401 <a href="#">__gnu_cxx::encoding_state</a> Class Reference	2103
5.401.1 Detailed Description	2104
5.402 <a href="#">std::encoding_state</a> Class Reference	2104
5.402.1 Detailed Description	2105
5.403 <a href="#">__gnu_pbds::detail::entry_cmp&lt;_VTp, Cmp_Fn, _Alloc, No_Throw&gt;</a> Struct Template Reference	2105
5.403.1 Detailed Description	2105
5.404 <a href="#">__gnu_pbds::detail::entry_cmp&lt;_VTp, Cmp_Fn, _Alloc, false&gt;</a> Struct Template Reference	2105
5.404.1 Detailed Description	2105
5.405 <a href="#">__gnu_pbds::detail::entry_cmp&lt;_VTp, Cmp_Fn, _Alloc, true&gt;</a> Struct Template Reference	2105
5.405.1 Detailed Description	2105
5.405.2 Member Typedef Documentation	2106
5.406 <a href="#">__gnu_pbds::detail::entry_pred&lt;_VTp, Pred, _Alloc, No_Throw&gt;</a> Struct Template Reference	2106
5.406.1 Detailed Description	2106
5.407 <a href="#">__gnu_pbds::detail::entry_pred&lt;_VTp, Pred, _Alloc, false&gt;</a> Struct Template Reference	2106
5.407.1 Detailed Description	2106
5.408 <a href="#">__gnu_pbds::detail::entry_pred&lt;_VTp, Pred, _Alloc, true&gt;</a> Struct Template Reference	2106
5.408.1 Detailed Description	2106
5.409 <a href="#">__gnu_pbds::detail::eq_by_less&lt;Key, Cmp_Fn&gt;</a> Struct Template Reference	2107
5.409.1 Detailed Description	2107
5.410 <a href="#">__gnu_parallel::equal_split_tag</a> Struct Reference	2107
5.410.1 Detailed Description	2107
5.411 <a href="#">__gnu_cxx::equal_to&lt;_Tp&gt;</a> Struct Template Reference	2108
5.411.1 Detailed Description	2108
5.411.2 Member Typedef Documentation	2108
5.412 <a href="#">std::equal_to&lt;_Tp&gt;</a> Struct Template Reference	2109
5.412.1 Detailed Description	2109
5.412.2 Member Typedef Documentation	2109
5.413 <a href="#">std::ranges::equal_to</a> Struct Reference	2110
5.413.1 Detailed Description	2110
5.414 <a href="#">std::equal_to&lt;void&gt;</a> Struct Reference	2110
5.414.1 Detailed Description	2111
5.414.2 Member Typedef Documentation	2111

5.415 <a href="#">std::error_category Class Reference</a>	2112
5.415.1 Detailed Description	2112
5.415.2 Member Function Documentation	2112
5.416 <a href="#">std::error_code Class Reference</a>	2113
5.416.1 Detailed Description	2114
5.416.2 Constructor & Destructor Documentation	2114
5.416.3 Member Function Documentation	2114
5.417 <a href="#">std::error_condition Class Reference</a>	2114
5.417.1 Detailed Description	2115
5.417.2 Constructor & Destructor Documentation	2115
5.417.3 Member Function Documentation	2116
5.418 <a href="#">__gnu_parallel::exact_tag Struct Reference</a>	2116
5.418.1 Detailed Description	2117
5.418.2 Member Function Documentation	2117
5.419 <a href="#">std::exception Class Reference</a>	2117
5.419.1 Detailed Description	2118
5.419.2 Member Function Documentation	2119
5.420 <a href="#">std::__unspecified__::exception_ptr Class Reference</a>	2119
5.420.1 Detailed Description	2119
5.421 <a href="#">std::exception_ptr Class Reference</a>	2119
5.421.1 Detailed Description	2120
5.422 <a href="#">std::exponential_distribution&lt;_RealType&gt; Class Template Reference</a>	2120
5.422.1 Detailed Description	2121
5.422.2 Member Typedef Documentation	2121
5.422.3 Constructor & Destructor Documentation	2121
5.422.4 Member Function Documentation	2122
5.422.5 Friends And Related Symbol Documentation	2123
5.423 <a href="#">std::extent&lt;typename, _UInt&gt; Struct Template Reference</a>	2123
5.423.1 Detailed Description	2124
5.424 <a href="#">std::extreme_value_distribution&lt;_RealType&gt; Class Template Reference</a>	2124
5.424.1 Detailed Description	2125
5.424.2 Member Typedef Documentation	2125
5.424.3 Member Function Documentation	2125
5.424.4 Friends And Related Symbol Documentation	2126
5.425 <a href="#">std::locale::facet Class Reference</a>	2126
5.425.1 Detailed Description	2128
5.425.2 Constructor & Destructor Documentation	2128
5.426 <a href="#">std::filesystem::file_status Class Reference</a>	2129
5.426.1 Detailed Description	2129

5.427 <a href="#">std::experimental::filesystem::v1::filesystem_error Class Reference</a>	2129
5.427.1 Detailed Description	2130
5.427.2 Member Function Documentation	2130
5.428 <a href="#">std::filesystem::filesystem_error Class Reference</a>	2131
5.428.1 Detailed Description	2131
5.428.2 Member Function Documentation	2131
5.429 <a href="#">__gnu_parallel::find_tag Struct Reference</a>	2132
5.429.1 Detailed Description	2132
5.430 <a href="#">std::fisher_f_distribution&lt;_RealType&gt; Class Template Reference</a>	2132
5.430.1 Detailed Description	2133
5.430.2 Member Typedef Documentation	2133
5.430.3 Member Function Documentation	2134
5.430.4 Friends And Related Symbol Documentation	2134
5.431 <a href="#">__gnu_cxx::forced_error Struct Reference</a>	2135
5.431.1 Detailed Description	2136
5.431.2 Member Function Documentation	2136
5.432 <a href="#">std::forward_iterator_tag Struct Reference</a>	2136
5.432.1 Detailed Description	2137
5.433 <a href="#">std::__debug::forward_list&lt;_Tp, _Alloc&gt; Class Template Reference</a>	2137
5.433.1 Detailed Description	2140
5.434 <a href="#">std::forward_list&lt;_Tp, _Alloc&gt; Class Template Reference</a>	2140
5.434.1 Detailed Description	2142
5.434.2 Constructor & Destructor Documentation	2143
5.434.3 Member Function Documentation	2146
5.435 <a href="#">std::fpos&lt;_StateT&gt; Class Template Reference</a>	2158
5.435.1 Detailed Description	2159
5.435.2 Constructor & Destructor Documentation	2159
5.435.3 Member Function Documentation	2159
5.436 <a href="#">__gnu_cxx::free_list Class Reference</a>	2160
5.436.1 Detailed Description	2161
5.436.2 Member Function Documentation	2161
5.437 <a href="#">std::from_chars_result Struct Reference</a>	2162
5.437.1 Detailed Description	2162
5.438 <a href="#">std::front_insert_iterator&lt;_Container&gt; Class Template Reference</a>	2162
5.438.1 Detailed Description	2163
5.438.2 Member Typedef Documentation	2163
5.438.3 Constructor & Destructor Documentation	2164
5.438.4 Member Function Documentation	2164
5.439 <a href="#">std::function&lt;_Res(_ArgTypes...)&gt; Class Template Reference</a>	2165

5.439.1 Detailed Description	2166
5.439.2 Constructor & Destructor Documentation	2166
5.439.3 Member Function Documentation	2167
5.440 <code>std::function_ref&lt;_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)&gt;</code> Class Template Reference	2171
5.440.1 Detailed Description	2171
5.440.2 Constructor & Destructor Documentation	2172
5.440.3 Member Function Documentation	2173
5.441 <code>std::future&lt;_Res &gt;</code> Class Template Reference	2173
5.441.1 Detailed Description	2174
5.441.2 Constructor & Destructor Documentation	2174
5.441.3 Member Function Documentation	2174
5.442 <code>std::future&lt;_Res &amp; &gt;</code> Class Template Reference	2174
5.442.1 Detailed Description	2176
5.442.2 Constructor & Destructor Documentation	2176
5.442.3 Member Function Documentation	2176
5.443 <code>std::future&lt;void &gt;</code> Class Reference	2176
5.443.1 Detailed Description	2178
5.443.2 Constructor & Destructor Documentation	2178
5.443.3 Member Function Documentation	2178
5.444 <code>std::future_error</code> Class Reference	2178
5.444.1 Detailed Description	2179
5.444.2 Member Function Documentation	2179
5.445 <code>std::gamma_distribution&lt;_RealType &gt;</code> Class Template Reference	2179
5.445.1 Detailed Description	2180
5.445.2 Member Typedef Documentation	2181
5.445.3 Constructor & Destructor Documentation	2181
5.445.4 Member Function Documentation	2181
5.445.5 Friends And Related Symbol Documentation	2182
5.446 <code>std::geometric_distribution&lt;_IntType &gt;</code> Class Template Reference	2183
5.446.1 Detailed Description	2184
5.446.2 Member Typedef Documentation	2184
5.446.3 Member Function Documentation	2184
5.446.4 Friends And Related Symbol Documentation	2185
5.447 <code>__gnu_pbds::gp_hash_table&lt;Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize←_Policy, Store_Hash, _Alloc &gt;</code> Class Template Reference	2186
5.447.1 Detailed Description	2187
5.447.2 Constructor & Destructor Documentation	2187
5.448 <code>__gnu_pbds::gp_hash_tag</code> Struct Reference	2191
5.448.1 Detailed Description	2192

5.449 <a href="#">__gnu_pbds::detail::gp_ht_map&lt; Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe←_Fn, Probe_Fn, Resize_Policy &gt; Class Template Reference</a>	2192
5.449.1 Detailed Description	2194
5.449.2 Member Enumeration Documentation	2194
5.449.3 Member Function Documentation	2194
5.450 <a href="#">std::chrono::gps_clock Class Reference</a>	2196
5.450.1 Detailed Description	2196
5.451 <a href="#">std::greater&lt; _Tp &gt; Struct Template Reference</a>	2197
5.451.1 Detailed Description	2197
5.451.2 Member Typedef Documentation	2197
5.452 <a href="#">std::ranges::greater Struct Reference</a>	2198
5.452.1 Detailed Description	2198
5.453 <a href="#">std::greater&lt; void &gt; Struct Reference</a>	2198
5.453.1 Detailed Description	2199
5.453.2 Member Typedef Documentation	2199
5.454 <a href="#">std::greater_equal&lt; _Tp &gt; Struct Template Reference</a>	2199
5.454.1 Detailed Description	2200
5.454.2 Member Typedef Documentation	2200
5.455 <a href="#">std::ranges::greater_equal Struct Reference</a>	2201
5.455.1 Detailed Description	2201
5.456 <a href="#">std::greater_equal&lt; void &gt; Struct Reference</a>	2201
5.456.1 Detailed Description	2202
5.456.2 Member Typedef Documentation	2202
5.457 <a href="#">__gnu_cxx::random_condition::group_adjustor Struct Reference</a>	2202
5.457.1 Detailed Description	2202
5.458 <a href="#">__gnu_parallel::growing_blocks_tag Struct Reference</a>	2202
5.458.1 Detailed Description	2203
5.459 <a href="#">std::gslice Class Reference</a>	2203
5.459.1 Detailed Description	2203
5.460 <a href="#">std::gslice_array&lt; _Tp &gt; Class Template Reference</a>	2204
5.460.1 Detailed Description	2205
5.460.2 Member Function Documentation	2205
5.461 <a href="#">std::has_virtual_destructor&lt; _Tp &gt; Struct Template Reference</a>	2206
5.461.1 Detailed Description	2206
5.462 <a href="#">std::hash&lt; _Tp &gt; Struct Template Reference</a>	2206
5.462.1 Detailed Description	2207
5.463 <a href="#">std::hash&lt; __debug::bitset&lt; _Nb &gt; &gt; Struct Template Reference</a>	2207
5.463.1 Detailed Description	2207
5.464 <a href="#">std::hash&lt; __debug::vector&lt; bool, _Alloc &gt; &gt; Struct Template Reference</a>	2207

5.464.1 Detailed Description	2207
5.465 std::hash< __gnu_cxx::__u16vstring > Struct Reference	2207
5.465.1 Detailed Description	2208
5.466 std::hash< __gnu_cxx::__u32vstring > Struct Reference	2208
5.466.1 Detailed Description	2208
5.467 std::hash< __gnu_cxx::__vstring > Struct Reference	2208
5.467.1 Detailed Description	2208
5.468 std::hash< __gnu_cxx::__wvstring > Struct Reference	2208
5.468.1 Detailed Description	2208
5.469 std::hash< __gnu_cxx::throw_value_limit > Struct Reference	2209
5.469.1 Detailed Description	2209
5.469.2 Member Typedef Documentation	2209
5.470 std::hash< __gnu_cxx::throw_value_random > Struct Reference	2210
5.470.1 Detailed Description	2210
5.470.2 Member Typedef Documentation	2210
5.471 std::hash< __gnu_debug::basic_string< _CharT > > Struct Template Reference	2211
5.471.1 Detailed Description	2211
5.472 std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference	2211
5.472.1 Detailed Description	2211
5.473 std::hash< _Tp * > Struct Template Reference	2211
5.473.1 Detailed Description	2212
5.474 std::hash< basic_string< char, char_traits< char >, _Alloc > > Struct Template Reference	2212
5.474.1 Detailed Description	2212
5.475 std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > > Struct Template Reference	2212
5.475.1 Detailed Description	2212
5.476 std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > > Struct Template Reference	2212
5.476.1 Detailed Description	2213
5.477 std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > > Struct Template Reference	2213
5.477.1 Detailed Description	2213
5.478 std::hash< bool > Struct Reference	2213
5.478.1 Detailed Description	2213
5.479 std::hash< char > Struct Reference	2213
5.479.1 Detailed Description	2213
5.480 std::hash< char16_t > Struct Reference	2214
5.480.1 Detailed Description	2214
5.481 std::hash< char32_t > Struct Reference	2214
5.481.1 Detailed Description	2214
5.482 std::hash< double > Struct Reference	2214
5.482.1 Detailed Description	2214

5.483 std::hash< error_code > Struct Reference . . . . .	2214
5.483.1 Detailed Description . . . . .	2214
5.484 std::hash< error_condition > Struct Reference . . . . .	2215
5.484.1 Detailed Description . . . . .	2215
5.485 std::hash< experimental::optional< _Tp > > Struct Template Reference . . . . .	2215
5.485.1 Detailed Description . . . . .	2215
5.486 std::hash< experimental::shared_ptr< _Tp > > Struct Template Reference . . . . .	2215
5.486.1 Detailed Description . . . . .	2215
5.487 std::hash< float > Struct Reference . . . . .	2216
5.487.1 Detailed Description . . . . .	2216
5.488 std::hash< int > Struct Reference . . . . .	2216
5.488.1 Detailed Description . . . . .	2216
5.489 std::hash< long > Struct Reference . . . . .	2216
5.489.1 Detailed Description . . . . .	2216
5.490 std::hash< long double > Struct Reference . . . . .	2216
5.490.1 Detailed Description . . . . .	2216
5.491 std::hash< long long > Struct Reference . . . . .	2217
5.491.1 Detailed Description . . . . .	2217
5.492 std::hash< shared_ptr< _Tp > > Struct Template Reference . . . . .	2217
5.492.1 Detailed Description . . . . .	2217
5.493 std::hash< short > Struct Reference . . . . .	2217
5.493.1 Detailed Description . . . . .	2217
5.494 std::hash< signed char > Struct Reference . . . . .	2217
5.494.1 Detailed Description . . . . .	2218
5.495 std::hash< thread::id > Struct Reference . . . . .	2218
5.495.1 Detailed Description . . . . .	2218
5.496 std::hash< type_index > Struct Reference . . . . .	2218
5.496.1 Detailed Description . . . . .	2218
5.497 std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference . . . . .	2218
5.497.1 Detailed Description . . . . .	2218
5.498 std::hash< unsigned char > Struct Reference . . . . .	2219
5.498.1 Detailed Description . . . . .	2219
5.499 std::hash< unsigned int > Struct Reference . . . . .	2219
5.499.1 Detailed Description . . . . .	2219
5.500 std::hash< unsigned long > Struct Reference . . . . .	2219
5.500.1 Detailed Description . . . . .	2219
5.501 std::hash< unsigned long long > Struct Reference . . . . .	2219
5.501.1 Detailed Description . . . . .	2219
5.502 std::hash< unsigned short > Struct Reference . . . . .	2220

5.502.1 Detailed Description	2220
5.503 std::hash< wchar_t > Struct Reference	2220
5.503.1 Detailed Description	2220
5.504 std::hash<::bitset< _Nb > > Struct Template Reference	2220
5.504.1 Detailed Description	2220
5.505 std::hash<::vector< bool, _Alloc > > Struct Template Reference	2220
5.505.1 Detailed Description	2221
5.506 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash > Struct Template Reference	2221
5.506.1 Detailed Description	2221
5.507 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false > Struct Template Reference	2222
5.507.1 Detailed Description	2222
5.508 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true > Struct Template Reference	2222
5.508.1 Detailed Description	2222
5.509 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference	2223
5.509.1 Detailed Description	2223
5.509.2 Constructor & Destructor Documentation	2223
5.510 __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference	2224
5.510.1 Detailed Description	2224
5.510.2 Member Enumeration Documentation	2225
5.510.3 Constructor & Destructor Documentation	2225
5.510.4 Member Function Documentation	2225
5.511 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference	2226
5.511.1 Detailed Description	2226
5.512 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true > Class Template Reference	2226
5.512.1 Detailed Description	2227
5.513 __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference	2227
5.513.1 Detailed Description	2228
5.514 __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference	2229
5.514.1 Detailed Description	2230
5.515 __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference	2230
5.515.1 Detailed Description	2231
5.516 __gnu_pbds::hash_prime_size_policy Class Reference	2232
5.516.1 Detailed Description	2232
5.516.2 Member Typedef Documentation	2232
5.516.3 Constructor & Destructor Documentation	2232
5.517 __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference	2232
5.517.1 Detailed Description	2234



5.518	<a href="#">__gnu_pbds::hash_standard_resize_policy&lt; Size_Policy, Trigger_Policy, External_Size_Access, Size_Type &gt; Class Template Reference</a>	2234
5.518.1	Detailed Description	2235
5.518.2	Member Enumeration Documentation	2235
5.518.3	Constructor & Destructor Documentation	2236
5.518.4	Member Function Documentation	2236
5.519	<a href="#">std::chrono::hh_mm_ss&lt; _Duration &gt; Class Template Reference</a>	2238
5.519.1	Detailed Description	2238
5.520	<a href="#">std::locale::id Class Reference</a>	2238
5.520.1	Detailed Description	2239
5.520.2	Constructor & Destructor Documentation	2239
5.520.3	Friends And Related Symbol Documentation	2239
5.521	<a href="#">std::thread::id Class Reference</a>	2240
5.521.1	Detailed Description	2240
5.522	<a href="#">std::identity Struct Reference</a>	2241
5.522.1	Detailed Description	2241
5.523	<a href="#">std::experimental::fundamentals_v1::in_place_t Struct Reference</a>	2241
5.523.1	Detailed Description	2241
5.524	<a href="#">std::independent_bits_engine&lt; _RandomNumberEngine, __w, _UIntType &gt; Class Template Reference</a>	2241
5.524.1	Detailed Description	2242
5.524.2	Member Typedef Documentation	2242
5.524.3	Constructor & Destructor Documentation	2242
5.524.4	Member Function Documentation	2243
5.524.5	Friends And Related Symbol Documentation	2244
5.525	<a href="#">std::indirect_array&lt; _Tp &gt; Class Template Reference</a>	2245
5.525.1	Detailed Description	2246
5.525.2	Member Function Documentation	2247
5.526	<a href="#">std::initializer_list&lt; _E &gt; Class Template Reference</a>	2248
5.526.1	Detailed Description	2248
5.526.2	Friends And Related Symbol Documentation	2249
5.527	<a href="#">std::input_iterator_tag Struct Reference</a>	2249
5.527.1	Detailed Description	2250
5.528	<a href="#">__gnu_pbds::insert_error Struct Reference</a>	2250
5.528.1	Detailed Description	2251
5.528.2	Member Function Documentation	2251
5.529	<a href="#">std::insert_iterator&lt; _Container &gt; Class Template Reference</a>	2251
5.529.1	Detailed Description	2252
5.529.2	Member Typedef Documentation	2253
5.529.3	Constructor & Destructor Documentation	2253

5.529.4 Member Function Documentation . . . . .	2253
5.530 std::integer_sequence< _Tp, _Idx > Struct Template Reference . . . . .	2254
5.530.1 Detailed Description . . . . .	2254
5.531 std::integral_constant< _Tp, __v > Struct Template Reference . . . . .	2255
5.531.1 Detailed Description . . . . .	2255
5.532 std::invalid_argument Class Reference . . . . .	2255
5.532.1 Detailed Description . . . . .	2256
5.532.2 Member Function Documentation . . . . .	2256
5.533 std::ios_base Class Reference . . . . .	2256
5.533.1 Detailed Description . . . . .	2259
5.533.2 Member Typedef Documentation . . . . .	2259
5.533.3 Member Enumeration Documentation . . . . .	2260
5.533.4 Constructor & Destructor Documentation . . . . .	2261
5.533.5 Member Function Documentation . . . . .	2261
5.533.6 Member Data Documentation . . . . .	2266
5.534 std::is_abstract< _Tp > Struct Template Reference . . . . .	2270
5.534.1 Detailed Description . . . . .	2270
5.535 std::is_arithmetic< _Tp > Struct Template Reference . . . . .	2270
5.535.1 Detailed Description . . . . .	2270
5.536 std::is_array< _Tp > Struct Template Reference . . . . .	2270
5.536.1 Detailed Description . . . . .	2271
5.537 std::is_assignable< _Tp, _Up > Struct Template Reference . . . . .	2271
5.537.1 Detailed Description . . . . .	2271
5.538 std::is_base_of< _Base, _Derived > Struct Template Reference . . . . .	2271
5.538.1 Detailed Description . . . . .	2271
5.539 std::is_bind_expression< _Tp > Struct Template Reference . . . . .	2271
5.539.1 Detailed Description . . . . .	2271
5.540 std::is_bind_expression< _Bind< _Signature > > Struct Template Reference . . . . .	2272
5.540.1 Detailed Description . . . . .	2272
5.541 std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct Template Reference . . . . .	2272
5.541.1 Detailed Description . . . . .	2272
5.542 std::is_bind_expression< const _Bind< _Signature > > Struct Template Reference . . . . .	2272
5.542.1 Detailed Description . . . . .	2272
5.543 std::is_bind_expression< const _Bind_result< _Result, _Signature > > Struct Template Reference . . . . .	2272
5.543.1 Detailed Description . . . . .	2272
5.544 std::is_bind_expression< const volatile _Bind< _Signature > > Struct Template Reference . . . . .	2273
5.544.1 Detailed Description . . . . .	2273
5.545 std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > > Struct Template Reference . . . . .	2273

5.545.1 Detailed Description	2273
5.546 std::is_bind_expression< volatile _Bind< _Signature > > Struct Template Reference	2273
5.546.1 Detailed Description	2273
5.547 std::is_bind_expression< volatile _Bind_result< _Result, _Signature > > Struct Template Reference	2273
5.547.1 Detailed Description	2273
5.548 std::is_class< _Tp > Struct Template Reference	2274
5.548.1 Detailed Description	2274
5.549 std::is_compound< _Tp > Struct Template Reference	2274
5.549.1 Detailed Description	2274
5.550 std::is_const< _Tp > Struct Template Reference	2274
5.550.1 Detailed Description	2274
5.551 std::is_constructible< _Tp, _Args > Struct Template Reference	2274
5.551.1 Detailed Description	2274
5.552 std::is_copy_assignable< _Tp > Struct Template Reference	2274
5.552.1 Detailed Description	2275
5.553 std::is_copy_constructible< _Tp > Struct Template Reference	2275
5.553.1 Detailed Description	2275
5.554 std::is_default_constructible< _Tp > Struct Template Reference	2275
5.554.1 Detailed Description	2275
5.555 std::is_destructible< _Tp > Struct Template Reference	2275
5.555.1 Detailed Description	2275
5.556 std::is_empty< _Tp > Struct Template Reference	2275
5.556.1 Detailed Description	2276
5.557 std::is_enum< _Tp > Struct Template Reference	2276
5.557.1 Detailed Description	2276
5.558 std::is_error_code_enum< _Tp > Struct Template Reference	2276
5.558.1 Detailed Description	2276
5.559 std::is_error_code_enum< future_errc > Struct Reference	2276
5.559.1 Detailed Description	2276
5.560 std::is_error_condition_enum< _Tp > Struct Template Reference	2276
5.560.1 Detailed Description	2277
5.561 std::is_floating_point< _Tp > Struct Template Reference	2277
5.561.1 Detailed Description	2277
5.562 std::is_function< _Tp > Struct Template Reference	2277
5.562.1 Detailed Description	2277
5.563 std::is_fundamental< _Tp > Struct Template Reference	2278
5.563.1 Detailed Description	2278
5.564 std::is_integral< _Tp > Struct Template Reference	2278
5.564.1 Detailed Description	2278

5.565 <code>std::is_layout_compatible&lt; _Tp, _Up &gt;</code> Struct Template Reference . . . . .	2278
5.565.1 Detailed Description . . . . .	2278
5.566 <code>std::is_literal_type&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2278
5.566.1 Detailed Description . . . . .	2278
5.567 <code>std::is_lvalue_reference&lt; typename &gt;</code> Struct Template Reference . . . . .	2279
5.567.1 Detailed Description . . . . .	2279
5.568 <code>std::is_member_function_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2279
5.568.1 Detailed Description . . . . .	2279
5.569 <code>std::is_member_object_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2279
5.569.1 Detailed Description . . . . .	2279
5.570 <code>std::is_member_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2279
5.570.1 Detailed Description . . . . .	2279
5.571 <code>std::is_move_assignable&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2280
5.571.1 Detailed Description . . . . .	2280
5.572 <code>std::is_move_constructible&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2280
5.572.1 Detailed Description . . . . .	2280
5.573 <code>std::is_nothrow_assignable&lt; _Tp, _Up &gt;</code> Struct Template Reference . . . . .	2280
5.573.1 Detailed Description . . . . .	2280
5.574 <code>std::is_nothrow_constructible&lt; _Tp, _Args &gt;</code> Struct Template Reference . . . . .	2280
5.574.1 Detailed Description . . . . .	2280
5.575 <code>std::is_nothrow_copy_assignable&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2280
5.575.1 Detailed Description . . . . .	2281
5.576 <code>std::is_nothrow_copy_constructible&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2281
5.576.1 Detailed Description . . . . .	2281
5.577 <code>std::is_nothrow_default_constructible&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2281
5.577.1 Detailed Description . . . . .	2281
5.578 <code>std::is_nothrow_destructible&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2281
5.578.1 Detailed Description . . . . .	2281
5.579 <code>std::is_nothrow_move_assignable&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2281
5.579.1 Detailed Description . . . . .	2282
5.580 <code>std::is_nothrow_move_constructible&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2282
5.580.1 Detailed Description . . . . .	2282
5.581 <code>std::is_object&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2282
5.581.1 Detailed Description . . . . .	2282
5.582 <code>std::is_placeholder&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2282
5.582.1 Detailed Description . . . . .	2283
5.583 <code>std::is_placeholder&lt; _Placeholder&lt; _Num &gt; &gt;</code> Struct Template Reference . . . . .	2283
5.583.1 Detailed Description . . . . .	2284
5.584 <code>std::is_pod&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2284

5.584.1 Detailed Description	2284
5.585 std::is_pointer< _Tp > Struct Template Reference	2285
5.585.1 Detailed Description	2285
5.586 std::is_pointer_interconvertible_base_of< _Base, _Derived > Struct Template Reference	2285
5.586.1 Detailed Description	2285
5.587 std::is_polymorphic< _Tp > Struct Template Reference	2285
5.587.1 Detailed Description	2285
5.588 std::is_reference< _Tp > Struct Template Reference	2285
5.588.1 Detailed Description	2286
5.589 std::is_rvalue_reference< typename > Struct Template Reference	2286
5.589.1 Detailed Description	2286
5.590 std::is_same< _Tp, _Up > Struct Template Reference	2286
5.590.1 Detailed Description	2286
5.591 std::is_scalar< _Tp > Struct Template Reference	2286
5.591.1 Detailed Description	2286
5.592 std::is_signed< _Tp > Struct Template Reference	2286
5.592.1 Detailed Description	2287
5.593 std::is_standard_layout< _Tp > Struct Template Reference	2287
5.593.1 Detailed Description	2287
5.594 std::is_trivial< _Tp > Struct Template Reference	2287
5.594.1 Detailed Description	2287
5.595 std::is_trivially_assignable< _Tp, _Up > Struct Template Reference	2287
5.595.1 Detailed Description	2287
5.596 std::is_trivially_constructible< _Tp, _Args > Struct Template Reference	2287
5.596.1 Detailed Description	2288
5.597 std::is_trivially_copy_assignable< _Tp > Struct Template Reference	2288
5.597.1 Detailed Description	2288
5.598 std::is_trivially_copy_constructible< _Tp > Struct Template Reference	2288
5.598.1 Detailed Description	2288
5.599 std::is_trivially_copyable< _Tp > Struct Template Reference	2288
5.599.1 Detailed Description	2288
5.600 std::is_trivially_default_constructible< _Tp > Struct Template Reference	2288
5.600.1 Detailed Description	2289
5.601 std::is_trivially_destructible< _Tp > Struct Template Reference	2289
5.601.1 Detailed Description	2289
5.602 std::is_trivially_move_assignable< _Tp > Struct Template Reference	2289
5.602.1 Detailed Description	2289
5.603 std::is_trivially_move_constructible< _Tp > Struct Template Reference	2289
5.603.1 Detailed Description	2289

5.604 std::is_union< _Tp > Struct Template Reference . . . . .	2289
5.604.1 Detailed Description . . . . .	2290
5.605 std::is_unsigned< _Tp > Struct Template Reference . . . . .	2290
5.605.1 Detailed Description . . . . .	2290
5.606 std::is_void< _Tp > Struct Template Reference . . . . .	2290
5.606.1 Detailed Description . . . . .	2290
5.607 std::is_volatile< _Tp > Struct Template Reference . . . . .	2290
5.607.1 Detailed Description . . . . .	2290
5.608 std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference . . . . .	2290
5.608.1 Detailed Description . . . . .	2291
5.608.2 Member Typedef Documentation . . . . .	2292
5.608.3 Constructor & Destructor Documentation . . . . .	2292
5.608.4 Friends And Related Symbol Documentation . . . . .	2292
5.609 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference . . . . .	2293
5.609.1 Detailed Description . . . . .	2294
5.609.2 Member Typedef Documentation . . . . .	2294
5.609.3 Constructor & Destructor Documentation . . . . .	2295
5.609.4 Member Function Documentation . . . . .	2296
5.610 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator Struct Reference . . . . .	2296
5.610.1 Detailed Description . . . . .	2298
5.611 std::experimental::filesystem::v1::path::iterator Class Reference . . . . .	2298
5.611.1 Detailed Description . . . . .	2298
5.612 std::filesystem::path::iterator Class Reference . . . . .	2298
5.612.1 Detailed Description . . . . .	2299
5.613 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference . . . . .	2299
5.613.1 Detailed Description . . . . .	2300
5.613.2 Member Typedef Documentation . . . . .	2300
5.614 std::iterator_traits< _Iterator > Struct Template Reference . . . . .	2301
5.614.1 Detailed Description . . . . .	2301
5.615 std::iterator_traits< _Tp * > Struct Template Reference . . . . .	2301
5.615.1 Detailed Description . . . . .	2301
5.616 __gnu_pbds::join_error Struct Reference . . . . .	2301
5.616.1 Detailed Description . . . . .	2302
5.616.2 Member Function Documentation . . . . .	2302
5.617 __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > Class Template Reference . . . . .	2302
5.617.1 Detailed Description . . . . .	2304
5.618 __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc > Class Template Ref- erence . . . . .	2304
5.618.1 Detailed Description . . . . .	2305

5.618.2 Member Typedef Documentation	2305
5.618.3 Constructor & Destructor Documentation	2306
5.618.4 Member Function Documentation	2306
5.619 <a href="#">__gnu_pbds::detail::left_child_next_sibling_heap_node&lt; _Value, _Metadata, _Alloc &gt; Struct Template Reference</a>	2307
5.619.1 Detailed Description	2308
5.620 <a href="#">__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator&lt; Node, _Alloc &gt; Class Template Reference</a>	2308
5.620.1 Detailed Description	2309
5.620.2 Member Typedef Documentation	2309
5.620.3 Constructor & Destructor Documentation	2310
5.620.4 Member Function Documentation	2311
5.621 <a href="#">std::length_error Class Reference</a>	2311
5.621.1 Detailed Description	2312
5.621.2 Member Function Documentation	2312
5.622 <a href="#">std::less&lt; _Tp &gt; Struct Template Reference</a>	2312
5.622.1 Detailed Description	2313
5.622.2 Member Typedef Documentation	2313
5.623 <a href="#">std::ranges::less Struct Reference</a>	2314
5.623.1 Detailed Description	2314
5.624 <a href="#">std::less&lt; void &gt; Struct Reference</a>	2314
5.624.1 Detailed Description	2315
5.624.2 Member Typedef Documentation	2315
5.625 <a href="#">std::less_equal&lt; _Tp &gt; Struct Template Reference</a>	2315
5.625.1 Detailed Description	2316
5.625.2 Member Typedef Documentation	2316
5.626 <a href="#">std::ranges::less_equal Struct Reference</a>	2316
5.626.1 Detailed Description	2316
5.627 <a href="#">std::less_equal&lt; void &gt; Struct Reference</a>	2317
5.627.1 Detailed Description	2317
5.627.2 Member Typedef Documentation	2317
5.628 <a href="#">__gnu_cxx::limit_condition::limit_adjustor Struct Reference</a>	2318
5.628.1 Detailed Description	2318
5.629 <a href="#">__gnu_cxx::limit_condition Struct Reference</a>	2318
5.629.1 Detailed Description	2318
5.630 <a href="#">std::linear_congruential_engine&lt; _UIntType, __a, __c, __m &gt; Class Template Reference</a>	2319
5.630.1 Detailed Description	2319
5.630.2 Member Typedef Documentation	2320
5.630.3 Constructor & Destructor Documentation	2320
5.630.4 Member Function Documentation	2321

5.630.5 Friends And Related Symbol Documentation . . . . .	2322
5.630.6 Member Data Documentation . . . . .	2323
5.631 <code>__gnu_pbds::linear_probe_fn&lt; Size_Type &gt;</code> Class Template Reference . . . . .	2323
5.631.1 Detailed Description . . . . .	2324
5.631.2 Member Function Documentation . . . . .	2324
5.632 <code>std::__debug::list&lt; _Tp, _Allocator &gt;</code> Class Template Reference . . . . .	2324
5.632.1 Detailed Description . . . . .	2326
5.633 <code>std::list&lt; _Tp, _Alloc &gt;</code> Class Template Reference . . . . .	2327
5.633.1 Detailed Description . . . . .	2330
5.633.2 Constructor & Destructor Documentation . . . . .	2331
5.633.3 Member Function Documentation . . . . .	2333
5.634 <code>__gnu_pbds::list_update&lt; Key, Mapped, Eq_Fn, Update_Policy, _Alloc &gt;</code> Class Template Reference . . . . .	2347
5.634.1 Detailed Description . . . . .	2347
5.634.2 Constructor & Destructor Documentation . . . . .	2348
5.635 <code>__gnu_pbds::list_update_tag</code> Struct Reference . . . . .	2348
5.635.1 Detailed Description . . . . .	2348
5.636 <code>std::locale</code> Class Reference . . . . .	2348
5.636.1 Detailed Description . . . . .	2350
5.636.2 Member Typedef Documentation . . . . .	2350
5.636.3 Constructor & Destructor Documentation . . . . .	2350
5.636.4 Member Function Documentation . . . . .	2353
5.636.5 Friends And Related Symbol Documentation . . . . .	2355
5.636.6 Member Data Documentation . . . . .	2356
5.637 <code>std::lock_guard&lt; _Mutex &gt;</code> Class Template Reference . . . . .	2357
5.637.1 Detailed Description . . . . .	2357
5.638 <code>std::logic_error</code> Class Reference . . . . .	2358
5.638.1 Detailed Description . . . . .	2358
5.638.2 Constructor & Destructor Documentation . . . . .	2358
5.638.3 Member Function Documentation . . . . .	2359
5.639 <code>std::logical_and&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2359
5.639.1 Detailed Description . . . . .	2359
5.639.2 Member Typedef Documentation . . . . .	2359
5.640 <code>std::logical_and&lt; void &gt;</code> Struct Reference . . . . .	2360
5.640.1 Detailed Description . . . . .	2360
5.640.2 Member Typedef Documentation . . . . .	2361
5.641 <code>std::logical_not&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2361
5.641.1 Detailed Description . . . . .	2361
5.641.2 Member Typedef Documentation . . . . .	2362
5.642 <code>std::logical_not&lt; void &gt;</code> Struct Reference . . . . .	2362



5.642.1 Detailed Description	2362
5.642.2 Member Typedef Documentation	2363
5.643 std::logical_or< _Tp > Struct Template Reference	2363
5.643.1 Detailed Description	2363
5.643.2 Member Typedef Documentation	2364
5.644 std::logical_or< void > Struct Reference	2364
5.644.1 Detailed Description	2365
5.644.2 Member Typedef Documentation	2365
5.645 std::lognormal_distribution< _RealType > Class Template Reference	2365
5.645.1 Detailed Description	2366
5.645.2 Member Typedef Documentation	2366
5.645.3 Member Function Documentation	2366
5.645.4 Friends And Related Symbol Documentation	2367
5.646 __gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference	2368
5.646.1 Detailed Description	2368
5.647 __gnu_pbds::lu_counter_policy< Max_Count, _Alloc > Class Template Reference	2369
5.647.1 Detailed Description	2369
5.647.2 Member Typedef Documentation	2369
5.647.3 Member Enumeration Documentation	2370
5.647.4 Member Function Documentation	2370
5.648 __gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference	2370
5.648.1 Detailed Description	2371
5.649 __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > Class Template Reference	2371
5.649.1 Detailed Description	2374
5.650 __gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference	2374
5.650.1 Detailed Description	2374
5.650.2 Member Typedef Documentation	2374
5.650.3 Member Function Documentation	2374
5.651 std::make_signed< _Tp > Struct Template Reference	2375
5.651.1 Detailed Description	2375
5.652 std::make_unsigned< _Tp > Struct Template Reference	2375
5.652.1 Detailed Description	2375
5.653 __gnu_cxx::malloc_allocator< _Tp > Class Template Reference	2375
5.653.1 Detailed Description	2376
5.654 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	2376
5.654.1 Detailed Description	2379
5.655 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2379
5.655.1 Detailed Description	2382
5.655.2 Constructor & Destructor Documentation	2382

5.655.3 Member Function Documentation . . . . .	2385
5.656 std::mask_array< _Tp > Class Template Reference . . . . .	2404
5.656.1 Detailed Description . . . . .	2405
5.656.2 Member Function Documentation . . . . .	2405
5.657 __gnu_pbds::detail::mask_based_range_hashing< Size_Type > Class Template Reference . . . . .	2407
5.657.1 Detailed Description . . . . .	2407
5.658 std::match_results< _Bi_iter, _Alloc > Class Template Reference . . . . .	2408
5.658.1 Detailed Description . . . . .	2410
5.658.2 Constructor & Destructor Documentation . . . . .	2410
5.658.3 Member Function Documentation . . . . .	2411
5.659 __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference	2416
5.659.1 Detailed Description . . . . .	2417
5.660 __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash > Struct Template Reference	2418
5.660.1 Detailed Description . . . . .	2418
5.661 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	2418
5.661.1 Detailed Description . . . . .	2419
5.661.2 Member Typedef Documentation . . . . .	2419
5.662 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	2420
5.662.1 Detailed Description . . . . .	2420
5.662.2 Member Typedef Documentation . . . . .	2420
5.663 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference . . . . .	2421
5.663.1 Detailed Description . . . . .	2421
5.663.2 Member Typedef Documentation . . . . .	2421
5.664 std::mem_fun_t< _Ret, _Tp > Class Template Reference . . . . .	2422
5.664.1 Detailed Description . . . . .	2422
5.664.2 Member Typedef Documentation . . . . .	2422
5.665 std::pmr::memory_resource Class Reference . . . . .	2423
5.665.1 Detailed Description . . . . .	2423
5.666 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference . . . . .	2423
5.666.1 Detailed Description . . . . .	2425
5.666.2 Member Typedef Documentation . . . . .	2425
5.666.3 Constructor & Destructor Documentation . . . . .	2426
5.666.4 Member Function Documentation . . . . .	2426
5.666.5 Friends And Related Symbol Documentation . . . . .	2427
5.667 std::messages< _CharT > Class Template Reference . . . . .	2428
5.667.1 Detailed Description . . . . .	2429
5.667.2 Member Typedef Documentation . . . . .	2430
5.667.3 Constructor & Destructor Documentation . . . . .	2430

5.667.4 Member Function Documentation	2431
5.667.5 Member Data Documentation	2431
5.668 std::messages_base Struct Reference	2431
5.668.1 Detailed Description	2432
5.669 std::messages_byname< _CharT > Class Template Reference	2432
5.669.1 Detailed Description	2433
5.669.2 Member Function Documentation	2433
5.669.3 Member Data Documentation	2433
5.670 std::minus< _Tp > Struct Template Reference	2434
5.670.1 Detailed Description	2434
5.670.2 Member Typedef Documentation	2434
5.671 std::minus< void > Struct Reference	2435
5.671.1 Detailed Description	2435
5.671.2 Member Typedef Documentation	2435
5.672 __gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference	2436
5.672.1 Detailed Description	2436
5.673 std::modulus< _Tp > Struct Template Reference	2436
5.673.1 Detailed Description	2437
5.673.2 Member Typedef Documentation	2437
5.674 std::modulus< void > Struct Reference	2437
5.674.1 Detailed Description	2438
5.674.2 Member Typedef Documentation	2438
5.675 std::money_base Class Reference	2438
5.675.1 Detailed Description	2439
5.676 std::money_get< _CharT, _InIter > Class Template Reference	2440
5.676.1 Detailed Description	2441
5.676.2 Member Typedef Documentation	2441
5.676.3 Constructor & Destructor Documentation	2441
5.676.4 Member Function Documentation	2442
5.676.5 Member Data Documentation	2444
5.677 std::money_put< _CharT, _OutIter > Class Template Reference	2444
5.677.1 Detailed Description	2445
5.677.2 Member Typedef Documentation	2445
5.677.3 Constructor & Destructor Documentation	2446
5.677.4 Member Function Documentation	2446
5.677.5 Member Data Documentation	2448
5.678 std::moneypunct< _CharT, _Intl > Class Template Reference	2448
5.678.1 Detailed Description	2450
5.678.2 Member Typedef Documentation	2450

5.678.3 Constructor & Destructor Documentation	2451
5.678.4 Member Function Documentation	2452
5.678.5 Member Data Documentation	2457
5.679 std::moneypunct_byname< _CharT, _Intl > Class Template Reference	2457
5.679.1 Detailed Description	2459
5.679.2 Member Function Documentation	2459
5.679.3 Member Data Documentation	2465
5.680 std::pmr::monotonic_buffer_resource Class Reference	2465
5.680.1 Detailed Description	2466
5.680.2 Member Function Documentation	2466
5.681 std::move_iterator< _Iterator > Class Template Reference	2466
5.681.1 Detailed Description	2467
5.682 std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference	2468
5.682.1 Detailed Description	2468
5.682.2 Constructor & Destructor Documentation	2469
5.682.3 Member Function Documentation	2470
5.682.4 Friends And Related Symbol Documentation	2471
5.683 std::move_sentinel< _Sent > Class Template Reference	2472
5.683.1 Detailed Description	2472
5.684 std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	2472
5.684.1 Detailed Description	2475
5.685 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2475
5.685.1 Detailed Description	2478
5.685.2 Constructor & Destructor Documentation	2478
5.685.3 Member Function Documentation	2481
5.686 std::multiplies< _Tp > Struct Template Reference	2498
5.686.1 Detailed Description	2499
5.686.2 Member Typedef Documentation	2499
5.687 std::multiplies< void > Struct Reference	2499
5.687.1 Detailed Description	2500
5.687.2 Member Typedef Documentation	2500
5.688 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference	2500
5.688.1 Detailed Description	2503
5.689 std::multiset< _Key, _Compare, _Alloc > Class Template Reference	2503
5.689.1 Detailed Description	2505
5.689.2 Constructor & Destructor Documentation	2506
5.689.3 Member Function Documentation	2509
5.690 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference	2523

5.690.1 Detailed Description	2523
5.690.2 Member Function Documentation	2523
5.691 <a href="#">__gnu_parallel::multiway_mergesort_sampling_tag Struct Reference</a>	2524
5.691.1 Detailed Description	2524
5.691.2 Member Function Documentation	2524
5.692 <a href="#">__gnu_parallel::multiway_mergesort_tag Struct Reference</a>	2525
5.692.1 Detailed Description	2525
5.692.2 Member Function Documentation	2525
5.693 <a href="#">std::mutex Class Reference</a>	2526
5.693.1 Detailed Description	2526
5.694 <a href="#">std::negate&lt; _Tp &gt; Struct Template Reference</a>	2526
5.694.1 Detailed Description	2527
5.694.2 Member Typedef Documentation	2527
5.695 <a href="#">std::negate&lt; void &gt; Struct Reference</a>	2527
5.695.1 Detailed Description	2528
5.695.2 Member Typedef Documentation	2528
5.696 <a href="#">std::negative_binomial_distribution&lt; _IntType &gt; Class Template Reference</a>	2528
5.696.1 Detailed Description	2529
5.696.2 Member Typedef Documentation	2530
5.696.3 Member Function Documentation	2530
5.696.4 Friends And Related Symbol Documentation	2531
5.697 <a href="#">std::nested_exception Class Reference</a>	2532
5.697.1 Detailed Description	2532
5.697.2 Constructor & Destructor Documentation	2532
5.697.3 Member Function Documentation	2532
5.698 <a href="#">__gnu_cxx::limit_condition::never_adjustor Struct Reference</a>	2533
5.698.1 Detailed Description	2533
5.699 <a href="#">__gnu_cxx::random_condition::never_adjustor Struct Reference</a>	2533
5.699.1 Detailed Description	2533
5.700 <a href="#">__gnu_cxx::new_allocator&lt; _Tp &gt; Class Template Reference</a>	2533
5.700.1 Detailed Description	2534
5.701 <a href="#">__gnu_pbds::detail::no_throw_copies&lt; Key, Mapped &gt; Struct Template Reference</a>	2534
5.701.1 Detailed Description	2534
5.702 <a href="#">__gnu_pbds::detail::no_throw_copies&lt; Key, null_type &gt; Struct Template Reference</a>	2535
5.702.1 Detailed Description	2535
5.703 <a href="#">std::normal_distribution&lt; _RealType &gt; Class Template Reference</a>	2535
5.703.1 Detailed Description	2536
5.703.2 Member Typedef Documentation	2536
5.703.3 Constructor & Destructor Documentation	2536

5.703.4 Member Function Documentation . . . . .	2536
5.703.5 Friends And Related Symbol Documentation . . . . .	2538
5.704 std::not_equal_to< _Tp > Struct Template Reference . . . . .	2539
5.704.1 Detailed Description . . . . .	2539
5.704.2 Member Typedef Documentation . . . . .	2539
5.705 std::ranges::not_equal_to Struct Reference . . . . .	2540
5.705.1 Detailed Description . . . . .	2540
5.706 std::not_equal_to< void > Struct Reference . . . . .	2540
5.706.1 Detailed Description . . . . .	2541
5.706.2 Member Typedef Documentation . . . . .	2541
5.707 __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 > Struct Template Reference . . . . .	2541
5.707.1 Detailed Description . . . . .	2542
5.708 __gnu_pbds::null_type Struct Reference . . . . .	2542
5.708.1 Detailed Description . . . . .	2543
5.709 std::experimental::fundamentals_v1::nullopt_t Struct Reference . . . . .	2543
5.709.1 Detailed Description . . . . .	2544
5.710 std::num_get< _CharT, _InIter > Class Template Reference . . . . .	2544
5.710.1 Detailed Description . . . . .	2546
5.710.2 Member Typedef Documentation . . . . .	2546
5.710.3 Constructor & Destructor Documentation . . . . .	2546
5.710.4 Member Function Documentation . . . . .	2546
5.710.5 Member Data Documentation . . . . .	2559
5.711 std::num_put< _CharT, _OutIter > Class Template Reference . . . . .	2559
5.711.1 Detailed Description . . . . .	2561
5.711.2 Member Typedef Documentation . . . . .	2561
5.711.3 Constructor & Destructor Documentation . . . . .	2562
5.711.4 Member Function Documentation . . . . .	2562
5.711.5 Member Data Documentation . . . . .	2572
5.712 std::numeric_limits< _Tp > Struct Template Reference . . . . .	2572
5.712.1 Detailed Description . . . . .	2573
5.712.2 Member Function Documentation . . . . .	2573
5.712.3 Member Data Documentation . . . . .	2574
5.713 std::numeric_limits< bool > Struct Reference . . . . .	2577
5.713.1 Detailed Description . . . . .	2579
5.713.2 Member Function Documentation . . . . .	2579
5.713.3 Member Data Documentation . . . . .	2580
5.714 std::numeric_limits< char > Struct Reference . . . . .	2582
5.714.1 Detailed Description . . . . .	2584
5.714.2 Member Function Documentation . . . . .	2584

5.714.3 Member Data Documentation	2585
5.715 std::numeric_limits< char16_t > Struct Reference	2587
5.715.1 Detailed Description	2589
5.715.2 Member Function Documentation	2589
5.715.3 Member Data Documentation	2590
5.716 std::numeric_limits< char32_t > Struct Reference	2593
5.716.1 Detailed Description	2594
5.716.2 Member Function Documentation	2595
5.716.3 Member Data Documentation	2596
5.717 std::numeric_limits< double > Struct Reference	2598
5.717.1 Detailed Description	2600
5.717.2 Member Function Documentation	2600
5.717.3 Member Data Documentation	2601
5.718 std::numeric_limits< float > Struct Reference	2603
5.718.1 Detailed Description	2605
5.718.2 Member Function Documentation	2605
5.718.3 Member Data Documentation	2606
5.719 std::numeric_limits< int > Struct Reference	2608
5.719.1 Detailed Description	2610
5.719.2 Member Function Documentation	2610
5.719.3 Member Data Documentation	2611
5.720 std::numeric_limits< long > Struct Reference	2614
5.720.1 Detailed Description	2615
5.720.2 Member Function Documentation	2615
5.720.3 Member Data Documentation	2616
5.721 std::numeric_limits< long double > Struct Reference	2619
5.721.1 Detailed Description	2621
5.721.2 Member Function Documentation	2621
5.721.3 Member Data Documentation	2622
5.722 std::numeric_limits< long long > Struct Reference	2624
5.722.1 Detailed Description	2626
5.722.2 Member Function Documentation	2626
5.722.3 Member Data Documentation	2627
5.723 std::numeric_limits< short > Struct Reference	2629
5.723.1 Detailed Description	2631
5.723.2 Member Function Documentation	2631
5.723.3 Member Data Documentation	2632
5.724 std::numeric_limits< signed char > Struct Reference	2635
5.724.1 Detailed Description	2636

5.724.2 Member Function Documentation	2637
5.724.3 Member Data Documentation	2638
5.725 std::numeric_limits< unsigned char > Struct Reference	2640
5.725.1 Detailed Description	2642
5.725.2 Member Function Documentation	2642
5.725.3 Member Data Documentation	2643
5.726 std::numeric_limits< unsigned int > Struct Reference	2645
5.726.1 Detailed Description	2647
5.726.2 Member Function Documentation	2647
5.726.3 Member Data Documentation	2648
5.727 std::numeric_limits< unsigned long > Struct Reference	2650
5.727.1 Detailed Description	2652
5.727.2 Member Function Documentation	2652
5.727.3 Member Data Documentation	2653
5.728 std::numeric_limits< unsigned long long > Struct Reference	2656
5.728.1 Detailed Description	2657
5.728.2 Member Function Documentation	2658
5.728.3 Member Data Documentation	2659
5.729 std::numeric_limits< unsigned short > Struct Reference	2661
5.729.1 Detailed Description	2663
5.729.2 Member Function Documentation	2663
5.729.3 Member Data Documentation	2664
5.730 std::numeric_limits< wchar_t > Struct Reference	2666
5.730.1 Detailed Description	2668
5.730.2 Member Function Documentation	2668
5.730.3 Member Data Documentation	2669
5.731 std::num_punct< _CharT > Class Template Reference	2671
5.731.1 Detailed Description	2673
5.731.2 Member Typedef Documentation	2673
5.731.3 Constructor & Destructor Documentation	2673
5.731.4 Member Function Documentation	2674
5.731.5 Member Data Documentation	2677
5.732 std::num_punct_byname< _CharT > Class Template Reference	2677
5.732.1 Detailed Description	2678
5.732.2 Member Function Documentation	2678
5.732.3 Member Data Documentation	2681
5.733 __gnu_parallel::omp_loop_static_tag Struct Reference	2681
5.733.1 Detailed Description	2682
5.733.2 Member Function Documentation	2682



5.734 <a href="#">__gnu_parallel::omp_loop_tag Struct Reference</a>	2682
5.734.1 Detailed Description	2683
5.734.2 Member Function Documentation	2683
5.735 <a href="#">std::once_flag Struct Reference</a>	2683
5.735.1 Detailed Description	2683
5.735.2 Constructor & Destructor Documentation	2684
5.735.3 Member Function Documentation	2684
5.735.4 Friends And Related Symbol Documentation	2684
5.736 <a href="#">std::experimental::fundamentals_v1::optional&lt;_Tp&gt; Class Template Reference</a>	2684
5.736.1 Detailed Description	2685
5.737 <a href="#">std::ostream_iterator&lt;_Tp, _CharT, _Traits&gt; Class Template Reference</a>	2686
5.737.1 Detailed Description	2686
5.737.2 Member Typedef Documentation	2687
5.737.3 Constructor & Destructor Documentation	2688
5.737.4 Member Function Documentation	2688
5.738 <a href="#">std::experimental::fundamentals_v2::ostream_joiner&lt;_DelimT, _CharT, _Traits&gt; Class Template Reference</a>	2688
5.738.1 Detailed Description	2689
5.739 <a href="#">std::ostreambuf_iterator&lt;_CharT, _Traits&gt; Class Template Reference</a>	2689
5.739.1 Detailed Description	2690
5.739.2 Member Typedef Documentation	2690
5.739.3 Constructor & Destructor Documentation	2691
5.739.4 Member Function Documentation	2691
5.740 <a href="#">std::out_of_range Class Reference</a>	2692
5.740.1 Detailed Description	2693
5.740.2 Member Function Documentation	2693
5.741 <a href="#">std::output_iterator_tag Struct Reference</a>	2693
5.741.1 Detailed Description	2694
5.742 <a href="#">__gnu_pbds::detail::ov_tree_map&lt;Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc&gt; Class Template Reference</a>	2694
5.742.1 Detailed Description	2696
5.742.2 Member Function Documentation	2696
5.743 <a href="#">__gnu_pbds::detail::ov_tree_node_const_it&lt;Value_Type, Metadata_Type, _Alloc&gt; Class Template Reference</a>	2697
5.743.1 Detailed Description	2698
5.743.2 Member Function Documentation	2698
5.744 <a href="#">__gnu_pbds::detail::ov_tree_node_it&lt;Value_Type, Metadata_Type, _Alloc&gt; Class Template Reference</a>	2699
5.744.1 Detailed Description	2700
5.744.2 Member Function Documentation	2700
5.745 <a href="#">__gnu_pbds::ov_tree_tag Struct Reference</a>	2700

5.745.1 Detailed Description . . . . .	2701
5.746 std::overflow_error Class Reference . . . . .	2701
5.746.1 Detailed Description . . . . .	2702
5.746.2 Member Function Documentation . . . . .	2702
5.747 std::owner_less< _Tp > Struct Template Reference . . . . .	2702
5.747.1 Detailed Description . . . . .	2702
5.748 std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > > Struct Template Reference . . . . .	2703
5.748.1 Detailed Description . . . . .	2703
5.748.2 Member Typedef Documentation . . . . .	2703
5.749 std::owner_less< shared_ptr< _Tp > > Struct Template Reference . . . . .	2703
5.749.1 Detailed Description . . . . .	2704
5.749.2 Member Typedef Documentation . . . . .	2704
5.750 std::owner_less< void > Struct Reference . . . . .	2704
5.750.1 Detailed Description . . . . .	2704
5.750.2 Member Typedef Documentation . . . . .	2705
5.751 std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > > Struct Template Reference . . . . .	2705
5.751.1 Detailed Description . . . . .	2705
5.751.2 Member Typedef Documentation . . . . .	2705
5.752 std::owner_less< weak_ptr< _Tp > > Struct Template Reference . . . . .	2706
5.752.1 Detailed Description . . . . .	2706
5.752.2 Member Typedef Documentation . . . . .	2706
5.753 std::packaged_task< _Res(_ArgTypes...) > Class Template Reference . . . . .	2706
5.753.1 Detailed Description . . . . .	2707
5.754 __gnu_cxx::pair< _T1, _T2 > Struct Template Reference . . . . .	2707
5.754.1 Detailed Description . . . . .	2709
5.754.2 Member Typedef Documentation . . . . .	2709
5.754.3 Constructor & Destructor Documentation . . . . .	2709
5.754.4 Member Function Documentation . . . . .	2710
5.754.5 Member Data Documentation . . . . .	2710
5.755 std::pair< _T1, _T2 > Struct Template Reference . . . . .	2710
5.755.1 Detailed Description . . . . .	2712
5.755.2 Member Typedef Documentation . . . . .	2712
5.755.3 Constructor & Destructor Documentation . . . . .	2713
5.755.4 Member Function Documentation . . . . .	2714
5.755.5 Member Data Documentation . . . . .	2714
5.756 __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . . .	2714
5.756.1 Detailed Description . . . . .	2716
5.757 __gnu_pbds::pairing_heap_tag Struct Reference . . . . .	2717
5.757.1 Detailed Description . . . . .	2717

5.758 <a href="#">__gnu_parallel::parallel_tag Struct Reference</a>	2717
5.758.1 Detailed Description	2719
5.758.2 Constructor & Destructor Documentation	2719
5.758.3 Member Function Documentation	2719
5.759 <a href="#">std::bernoulli_distribution::param_type Struct Reference</a>	2719
5.759.1 Detailed Description	2720
5.760 <a href="#">std::binomial_distribution&lt;_IntType&gt;::param_type Struct Reference</a>	2720
5.760.1 Detailed Description	2720
5.761 <a href="#">std::cauchy_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2720
5.761.1 Detailed Description	2721
5.762 <a href="#">std::chi_squared_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2721
5.762.1 Detailed Description	2721
5.763 <a href="#">std::discrete_distribution&lt;_IntType&gt;::param_type Struct Reference</a>	2721
5.763.1 Detailed Description	2722
5.764 <a href="#">std::exponential_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2722
5.764.1 Detailed Description	2722
5.765 <a href="#">std::extreme_value_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2722
5.765.1 Detailed Description	2723
5.766 <a href="#">std::fisher_f_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2723
5.766.1 Detailed Description	2723
5.767 <a href="#">std::gamma_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2723
5.767.1 Detailed Description	2724
5.768 <a href="#">std::geometric_distribution&lt;_IntType&gt;::param_type Struct Reference</a>	2724
5.768.1 Detailed Description	2724
5.769 <a href="#">std::lognormal_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2724
5.769.1 Detailed Description	2725
5.770 <a href="#">std::negative_binomial_distribution&lt;_IntType&gt;::param_type Struct Reference</a>	2725
5.770.1 Detailed Description	2725
5.771 <a href="#">std::normal_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2725
5.771.1 Detailed Description	2726
5.772 <a href="#">std::piecewise_constant_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2726
5.772.1 Detailed Description	2727
5.773 <a href="#">std::piecewise_linear_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2727
5.773.1 Detailed Description	2727
5.774 <a href="#">std::poisson_distribution&lt;_IntType&gt;::param_type Struct Reference</a>	2727
5.774.1 Detailed Description	2728
5.775 <a href="#">std::student_t_distribution&lt;_RealType&gt;::param_type Struct Reference</a>	2728
5.775.1 Detailed Description	2728
5.776 <a href="#">std::uniform_int_distribution&lt;_IntType&gt;::param_type Struct Reference</a>	2728

5.776.1 Detailed Description	2729
5.777 <code>std::uniform_real_distribution&lt;_RealType&gt;::param_type</code> Struct Reference	2729
5.777.1 Detailed Description	2729
5.778 <code>std::weibull_distribution&lt;_RealType&gt;::param_type</code> Struct Reference	2729
5.778.1 Detailed Description	2730
5.779 <code>__gnu_pbds::detail::pat_trie_base</code> Struct Reference	2730
5.779.1 Detailed Description	2731
5.779.2 Member Enumeration Documentation	2731
5.780 <code>__gnu_pbds::detail::pat_trie_map&lt;Key, Mapped, Node_And_It_Traits, _Alloc&gt;</code> Class Template Reference	2731
5.780.1 Detailed Description	2733
5.780.2 Member Enumeration Documentation	2733
5.780.3 Member Function Documentation	2733
5.781 <code>__gnu_pbds::pat_trie_tag</code> Struct Reference	2734
5.781.1 Detailed Description	2735
5.782 <code>std::experimental::filesystem::v1::path</code> Class Reference	2735
5.782.1 Detailed Description	2737
5.783 <code>std::filesystem::path</code> Class Reference	2738
5.783.1 Detailed Description	2740
5.783.2 Member Enumeration Documentation	2740
5.783.3 Friends And Related Symbol Documentation	2741
5.784 <code>std::piecewise_constant_distribution&lt;_RealType&gt;</code> Class Template Reference	2741
5.784.1 Detailed Description	2742
5.784.2 Member Typedef Documentation	2743
5.784.3 Member Function Documentation	2743
5.784.4 Friends And Related Symbol Documentation	2745
5.785 <code>std::piecewise_construct_t</code> Struct Reference	2746
5.785.1 Detailed Description	2746
5.786 <code>std::piecewise_linear_distribution&lt;_RealType&gt;</code> Class Template Reference	2746
5.786.1 Detailed Description	2747
5.786.2 Member Typedef Documentation	2747
5.786.3 Member Function Documentation	2747
5.786.4 Friends And Related Symbol Documentation	2748
5.787 <code>std::plus&lt;_Tp&gt;</code> Struct Template Reference	2749
5.787.1 Detailed Description	2750
5.787.2 Member Typedef Documentation	2750
5.787.3 Member Function Documentation	2751
5.788 <code>__gnu_pbds::point_invalidation_guarantee</code> Struct Reference	2751
5.788.1 Detailed Description	2751

5.789 <a href="#">std::pointer_to_binary_function&lt; _Arg1, _Arg2, _Result &gt; Class Template Reference</a> . . . . .	2751
5.789.1 Detailed Description . . . . .	2752
5.789.2 Member Typedef Documentation . . . . .	2752
5.790 <a href="#">std::pointer_to_unary_function&lt; _Arg, _Result &gt; Class Template Reference</a> . . . . .	2753
5.790.1 Detailed Description . . . . .	2753
5.790.2 Member Typedef Documentation . . . . .	2754
5.791 <a href="#">std::pointer_traits&lt; _Ptr &gt; Struct Template Reference</a> . . . . .	2754
5.791.1 Detailed Description . . . . .	2754
5.791.2 Member Typedef Documentation . . . . .	2754
5.791.3 Member Function Documentation . . . . .	2755
5.792 <a href="#">std::pointer_traits&lt; _Tp * &gt; Struct Template Reference</a> . . . . .	2755
5.792.1 Detailed Description . . . . .	2756
5.792.2 Member Typedef Documentation . . . . .	2756
5.792.3 Member Function Documentation . . . . .	2757
5.793 <a href="#">std::poisson_distribution&lt; _IntType &gt; Class Template Reference</a> . . . . .	2758
5.793.1 Detailed Description . . . . .	2759
5.793.2 Member Typedef Documentation . . . . .	2759
5.793.3 Member Function Documentation . . . . .	2759
5.793.4 Friends And Related Symbol Documentation . . . . .	2760
5.794 <a href="#">std::pmr::polymorphic_allocator&lt; _Tp &gt; Class Template Reference</a> . . . . .	2761
5.794.1 Detailed Description . . . . .	2762
5.795 <a href="#">std::pmr::pool_options Struct Reference</a> . . . . .	2762
5.795.1 Detailed Description . . . . .	2762
5.795.2 Member Data Documentation . . . . .	2763
5.796 <a href="#">__gnu_pbds::priority_queue&lt; _Tv, Cmp_Fn, Tag, _Alloc &gt; Class Template Reference</a> . . . . .	2763
5.796.1 Detailed Description . . . . .	2763
5.796.2 Constructor & Destructor Documentation . . . . .	2764
5.797 <a href="#">std::priority_queue&lt; _Tp, _Sequence, _Compare &gt; Class Template Reference</a> . . . . .	2764
5.797.1 Detailed Description . . . . .	2766
5.797.2 Constructor & Destructor Documentation . . . . .	2766
5.797.3 Member Function Documentation . . . . .	2767
5.798 <a href="#">__gnu_pbds::priority_queue_tag Struct Reference</a> . . . . .	2768
5.798.1 Detailed Description . . . . .	2768
5.799 <a href="#">__gnu_pbds::detail::probe_fn_base&lt; _Alloc &gt; Class Template Reference</a> . . . . .	2769
5.799.1 Detailed Description . . . . .	2769
5.800 <a href="#">__gnu_cxx::project1st&lt; _Arg1, _Arg2 &gt; Struct Template Reference</a> . . . . .	2769
5.800.1 Detailed Description . . . . .	2769
5.800.2 Member Typedef Documentation . . . . .	2769
5.801 <a href="#">__gnu_cxx::project2nd&lt; _Arg1, _Arg2 &gt; Struct Template Reference</a> . . . . .	2769

5.801.1 Detailed Description	2770
5.801.2 Member Typedef Documentation	2770
5.802 std::promise< _Res > Class Template Reference	2770
5.802.1 Detailed Description	2771
5.803 std::promise< _Res & > Class Template Reference	2771
5.803.1 Detailed Description	2772
5.804 std::promise< void > Class Reference	2772
5.804.1 Detailed Description	2773
5.805 std::experimental::fundamentals_v2::propagate_const< _Tp > Class Template Reference	2773
5.805.1 Detailed Description	2774
5.806 __gnu_pbds::quadratic_probe_fn< Size_Type > Class Template Reference	2774
5.806.1 Detailed Description	2774
5.806.2 Member Function Documentation	2774
5.807 std::queue< _Tp, _Sequence > Class Template Reference	2774
5.807.1 Detailed Description	2776
5.807.2 Constructor & Destructor Documentation	2776
5.807.3 Member Function Documentation	2776
5.807.4 Member Data Documentation	2777
5.808 __gnu_parallel::quicksort_tag Struct Reference	2778
5.808.1 Detailed Description	2778
5.808.2 Member Function Documentation	2778
5.809 std::random_access_iterator_tag Struct Reference	2779
5.809.1 Detailed Description	2779
5.810 __gnu_cxx::random_condition Struct Reference	2779
5.810.1 Detailed Description	2780
5.811 std::random_device Class Reference	2780
5.811.1 Detailed Description	2781
5.811.2 Member Typedef Documentation	2781
5.812 std::range_error Class Reference	2781
5.812.1 Detailed Description	2781
5.812.2 Member Function Documentation	2782
5.813 __gnu_pbds::range_invalidation_guarantee Struct Reference	2782
5.813.1 Detailed Description	2782
5.814 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference	2783
5.814.1 Detailed Description	2783
5.815 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > Class Template Reference	2783
5.815.1 Detailed Description	2784

5.816 <a href="#">__gnu_pbds::detail::ranged_hash_fn&lt; Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true &gt; Class Template Reference</a>	2784
5.816.1 Detailed Description	2784
5.817 <a href="#">__gnu_pbds::detail::ranged_hash_fn&lt; Key, null_type, _Alloc, Comb_Hash_Fn, false &gt; Class Template Reference</a>	2784
5.817.1 Detailed Description	2785
5.818 <a href="#">__gnu_pbds::detail::ranged_hash_fn&lt; Key, null_type, _Alloc, Comb_Hash_Fn, true &gt; Class Template Reference</a>	2785
5.818.1 Detailed Description	2785
5.819 <a href="#">__gnu_pbds::detail::ranged_probe_fn&lt; Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_↵ Hash &gt; Class Template Reference</a>	2786
5.819.1 Detailed Description	2786
5.820 <a href="#">__gnu_pbds::detail::ranged_probe_fn&lt; Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false &gt; Class Template Reference</a>	2786
5.820.1 Detailed Description	2787
5.821 <a href="#">__gnu_pbds::detail::ranged_probe_fn&lt; Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true &gt; Class Template Reference</a>	2787
5.821.1 Detailed Description	2787
5.822 <a href="#">__gnu_pbds::detail::ranged_probe_fn&lt; Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false &gt; Class Template Reference</a>	2788
5.822.1 Detailed Description	2788
5.823 <a href="#">std::rank&lt; _Tp &gt; Struct Template Reference</a>	2788
5.823.1 Detailed Description	2789
5.824 <a href="#">std::ratio&lt; _Num, _Den &gt; Struct Template Reference</a>	2789
5.824.1 Detailed Description	2789
5.825 <a href="#">std::ratio_equal&lt; _R1, _R2 &gt; Struct Template Reference</a>	2789
5.825.1 Detailed Description	2790
5.826 <a href="#">std::ratio_greater&lt; _R1, _R2 &gt; Struct Template Reference</a>	2790
5.826.1 Detailed Description	2791
5.827 <a href="#">std::ratio_greater_equal&lt; _R1, _R2 &gt; Struct Template Reference</a>	2791
5.827.1 Detailed Description	2792
5.828 <a href="#">std::ratio_less&lt; _R1, _R2 &gt; Struct Template Reference</a>	2792
5.828.1 Detailed Description	2792
5.829 <a href="#">std::ratio_less_equal&lt; _R1, _R2 &gt; Struct Template Reference</a>	2793
5.829.1 Detailed Description	2793
5.830 <a href="#">std::ratio_not_equal&lt; _R1, _R2 &gt; Struct Template Reference</a>	2793
5.830.1 Detailed Description	2794
5.831 <a href="#">std::raw_storage_iterator&lt; _OutputIterator, _Tp &gt; Class Template Reference</a>	2794
5.831.1 Detailed Description	2795
5.831.2 Member Typedef Documentation	2795
5.832 <a href="#">__gnu_cxx::rb_tree&lt; _Key, _Value, _KeyOfValue, _Compare, _Alloc &gt; Struct Template Reference</a>	2796

5.832.1 Detailed Description	2799
5.833 <code>__gnu_pbds::detail::rb_tree_map&lt; Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc &gt; Class Template Reference</code>	2800
5.833.1 Detailed Description	2802
5.833.2 Member Function Documentation	2803
5.834 <code>__gnu_pbds::detail::rb_tree_node_&lt; Value_Type, Metadata, _Alloc &gt; Struct Template Reference</code>	2803
5.834.1 Detailed Description	2804
5.835 <code>__gnu_pbds::rb_tree_tag Struct Reference</code>	2804
5.835.1 Detailed Description	2805
5.836 <code>__gnu_pbds::detail::rc&lt; _Node, _Alloc &gt; Class Template Reference</code>	2805
5.836.1 Detailed Description	2806
5.837 <code>__gnu_pbds::detail::rc_binomial_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</code>	2806
5.837.1 Detailed Description	2808
5.838 <code>__gnu_pbds::rc_binomial_heap_tag Struct Reference</code>	2808
5.838.1 Detailed Description	2808
5.839 <code>__gnu_pbds::detail::rebind_traits&lt; _Alloc, T &gt; Struct Template Reference</code>	2808
5.839.1 Detailed Description	2809
5.840 <code>std::filesystem::recursive_directory_iterator Class Reference</code>	2809
5.840.1 Detailed Description	2810
5.841 <code>__gnu_cxx::recursive_init_error Class Reference</code>	2810
5.841.1 Detailed Description	2810
5.841.2 Member Function Documentation	2810
5.842 <code>std::recursive_mutex Class Reference</code>	2811
5.842.1 Detailed Description	2811
5.843 <code>std::recursive_timed_mutex Class Reference</code>	2811
5.843.1 Detailed Description	2812
5.844 <code>std::bitset&lt; _Nb &gt;::reference Class Reference</code>	2812
5.844.1 Detailed Description	2812
5.845 <code>std::tr2::dynamic_bitset&lt; _WordT, _Alloc &gt;::reference Class Reference</code>	2812
5.845.1 Detailed Description	2813
5.846 <code>std::reference_wrapper&lt; _Tp &gt; Class Template Reference</code>	2813
5.846.1 Detailed Description	2813
5.846.2 Friends And Related Symbol Documentation	2814
5.847 <code>std::regex_error Class Reference</code>	2814
5.847.1 Detailed Description	2815
5.847.2 Constructor & Destructor Documentation	2815
5.847.3 Member Function Documentation	2815
5.848 <code>std::regex_iterator&lt; _Bi_iter, _Ch_type, _Rx_traits &gt; Class Template Reference</code>	2816
5.848.1 Detailed Description	2816



5.848.2 Constructor & Destructor Documentation	2817
5.848.3 Member Function Documentation	2817
5.849 std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	2818
5.849.1 Detailed Description	2819
5.849.2 Constructor & Destructor Documentation	2819
5.849.3 Member Function Documentation	2822
5.850 std::regex_traits< _Ch_type > Class Template Reference	2823
5.850.1 Detailed Description	2823
5.850.2 Constructor & Destructor Documentation	2824
5.850.3 Member Function Documentation	2824
5.851 std::remove_all_extents< _Tp > Struct Template Reference	2828
5.851.1 Detailed Description	2829
5.852 std::remove_const< _Tp > Struct Template Reference	2829
5.852.1 Detailed Description	2829
5.853 std::remove_cv< _Tp > Struct Template Reference	2829
5.853.1 Detailed Description	2829
5.854 std::remove_extent< _Tp > Struct Template Reference	2829
5.854.1 Detailed Description	2830
5.855 std::remove_pointer< _Tp > Struct Template Reference	2830
5.855.1 Detailed Description	2830
5.856 std::remove_reference< _Tp > Struct Template Reference	2830
5.856.1 Detailed Description	2830
5.857 std::remove_volatile< _Tp > Struct Template Reference	2830
5.857.1 Detailed Description	2831
5.858 __gnu_pbds::resize_error Struct Reference	2831
5.858.1 Detailed Description	2831
5.858.2 Member Function Documentation	2831
5.859 __gnu_pbds::detail::resize_policy< _Tp > Class Template Reference	2832
5.859.1 Detailed Description	2832
5.860 std::result_of< _Signature > Struct Template Reference	2832
5.860.1 Detailed Description	2832
5.861 std::reverse_iterator< _Iterator > Class Template Reference	2833
5.861.1 Detailed Description	2834
5.861.2 Constructor & Destructor Documentation	2834
5.861.3 Member Function Documentation	2835
5.862 __gnu_cxx::rope< _CharT, _Alloc > Class Template Reference	2837
5.862.1 Detailed Description	2842
5.863 std::runtime_error Class Reference	2842
5.863.1 Detailed Description	2843

---

5.863.2 Constructor & Destructor Documentation	2843
5.863.3 Member Function Documentation	2843
5.864 <a href="#">__gnu_pbds::sample_probe_fn Class Reference</a>	2843
5.864.1 Detailed Description	2843
5.864.2 Constructor & Destructor Documentation	2844
5.864.3 Member Function Documentation	2844
5.865 <a href="#">__gnu_pbds::sample_range_hashing Class Reference</a>	2844
5.865.1 Detailed Description	2844
5.865.2 Member Typedef Documentation	2845
5.865.3 Constructor & Destructor Documentation	2845
5.865.4 Member Function Documentation	2845
5.866 <a href="#">__gnu_pbds::sample_ranged_hash_fn Class Reference</a>	2845
5.866.1 Detailed Description	2846
5.866.2 Constructor & Destructor Documentation	2846
5.866.3 Member Function Documentation	2846
5.867 <a href="#">__gnu_pbds::sample_ranged_probe_fn Class Reference</a>	2846
5.867.1 Detailed Description	2847
5.868 <a href="#">__gnu_pbds::sample_resize_policy Class Reference</a>	2847
5.868.1 Detailed Description	2848
5.868.2 Member Typedef Documentation	2848
5.868.3 Constructor & Destructor Documentation	2848
5.868.4 Member Function Documentation	2848
5.869 <a href="#">__gnu_pbds::sample_resize_trigger Class Reference</a>	2850
5.869.1 Detailed Description	2850
5.869.2 Member Typedef Documentation	2850
5.869.3 Constructor & Destructor Documentation	2850
5.869.4 Member Function Documentation	2851
5.870 <a href="#">__gnu_pbds::sample_size_policy Class Reference</a>	2852
5.870.1 Detailed Description	2853
5.870.2 Member Typedef Documentation	2853
5.870.3 Constructor & Destructor Documentation	2853
5.870.4 Member Function Documentation	2853
5.871 <a href="#">__gnu_pbds::sample_tree_node_update&lt; Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	2854
5.871.1 Detailed Description	2854
5.872 <a href="#">__gnu_pbds::sample_trie_access_traits Struct Reference</a>	2854
5.872.1 Detailed Description	2854
5.872.2 Member Typedef Documentation	2854
5.872.3 Member Function Documentation	2854

5.873 <a href="#">__gnu_pbds::sample_trie_node_update&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt; Class Template Reference</a>	2855
5.873.1 Detailed Description	2855
5.873.2 Constructor & Destructor Documentation	2855
5.873.3 Member Function Documentation	2855
5.874 <a href="#">__gnu_pbds::sample_update_policy Struct Reference</a>	2856
5.874.1 Detailed Description	2856
5.874.2 Member Typedef Documentation	2856
5.874.3 Constructor & Destructor Documentation	2856
5.874.4 Member Function Documentation	2856
5.875 <a href="#">__gnu_parallel::sampling_tag Struct Reference</a>	2857
5.875.1 Detailed Description	2857
5.875.2 Member Function Documentation	2857
5.876 <a href="#">std::scoped_allocator_adaptor&lt; _OuterAlloc, _InnerAllocs &gt; Class Template Reference</a>	2858
5.876.1 Detailed Description	2859
5.877 <a href="#">std::seed_seq Class Reference</a>	2859
5.877.1 Detailed Description	2860
5.877.2 Member Typedef Documentation	2860
5.877.3 Constructor & Destructor Documentation	2860
5.878 <a href="#">__gnu_cxx::select1st&lt; _Pair &gt; Struct Template Reference</a>	2860
5.878.1 Detailed Description	2861
5.878.2 Member Typedef Documentation	2861
5.879 <a href="#">__gnu_cxx::select2nd&lt; _Pair &gt; Struct Template Reference</a>	2861
5.879.1 Detailed Description	2861
5.879.2 Member Typedef Documentation	2861
5.880 <a href="#">__gnu_pbds::detail::select_value_type&lt; Key, Mapped &gt; Struct Template Reference</a>	2862
5.880.1 Detailed Description	2862
5.881 <a href="#">__gnu_pbds::detail::select_value_type&lt; Key, null_type &gt; Struct Template Reference</a>	2862
5.881.1 Detailed Description	2862
5.882 <a href="#">std::basic_istream&lt; _CharT, _Traits &gt;::sentry Class Reference</a>	2862
5.882.1 Detailed Description	2868
5.882.2 Member Typedef Documentation	2868
5.882.3 Member Enumeration Documentation	2869
5.882.4 Constructor & Destructor Documentation	2869
5.882.5 Member Function Documentation	2870
5.882.6 Member Data Documentation	2893
5.883 <a href="#">std::basic_ostream&lt; _CharT, _Traits &gt;::sentry Class Reference</a>	2897
5.883.1 Detailed Description	2901
5.883.2 Member Typedef Documentation	2901

5.883.3 Member Enumeration Documentation . . . . .	2902
5.883.4 Constructor & Destructor Documentation . . . . .	2903
5.883.5 Member Function Documentation . . . . .	2903
5.883.6 Member Data Documentation . . . . .	2921
5.884 <code>__gnu_pbds::sequence_tag</code> Struct Reference . . . . .	2924
5.884.1 Detailed Description . . . . .	2925
5.885 <code>__gnu_parallel::sequential_tag</code> Struct Reference . . . . .	2925
5.885.1 Detailed Description . . . . .	2925
5.886 <code>std::__debug::set&lt; _Key, _Compare, _Allocator &gt;</code> Class Template Reference . . . . .	2925
5.886.1 Detailed Description . . . . .	2928
5.887 <code>std::set&lt; _Key, _Compare, _Alloc &gt;</code> Class Template Reference . . . . .	2928
5.887.1 Detailed Description . . . . .	2931
5.887.2 Member Typedef Documentation . . . . .	2931
5.887.3 Constructor & Destructor Documentation . . . . .	2933
5.887.4 Member Function Documentation . . . . .	2936
5.888 <code>std::shared_future&lt; _Res &gt;</code> Class Template Reference . . . . .	2951
5.888.1 Detailed Description . . . . .	2952
5.888.2 Constructor & Destructor Documentation . . . . .	2952
5.888.3 Member Function Documentation . . . . .	2952
5.889 <code>std::shared_future&lt; _Res &amp; &gt;</code> Class Template Reference . . . . .	2953
5.889.1 Detailed Description . . . . .	2954
5.889.2 Constructor & Destructor Documentation . . . . .	2954
5.889.3 Member Function Documentation . . . . .	2954
5.890 <code>std::shared_future&lt; void &gt;</code> Class Reference . . . . .	2955
5.890.1 Detailed Description . . . . .	2956
5.890.2 Constructor & Destructor Documentation . . . . .	2956
5.890.3 Member Function Documentation . . . . .	2957
5.891 <code>std::shared_lock&lt; _Mutex &gt;</code> Class Template Reference . . . . .	2957
5.891.1 Detailed Description . . . . .	2958
5.892 <code>std::shared_ptr&lt; _Tp &gt;</code> Class Template Reference . . . . .	2958
5.892.1 Detailed Description . . . . .	2960
5.892.2 Member Typedef Documentation . . . . .	2960
5.892.3 Constructor & Destructor Documentation . . . . .	2960
5.892.4 Member Function Documentation . . . . .	2966
5.893 <code>std::shared_timed_mutex</code> Class Reference . . . . .	2967
5.893.1 Detailed Description . . . . .	2968
5.894 <code>std::shuffle_order_engine&lt; _RandomNumberEngine, __k &gt;</code> Class Template Reference . . . . .	2968
5.894.1 Detailed Description . . . . .	2969
5.894.2 Member Typedef Documentation . . . . .	2969

5.894.3 Constructor & Destructor Documentation	2969
5.894.4 Member Function Documentation	2970
5.894.5 Friends And Related Symbol Documentation	2971
5.895 std::slice Class Reference	2973
5.895.1 Detailed Description	2973
5.895.2 Friends And Related Symbol Documentation	2973
5.896 std::slice_array< _Tp > Class Template Reference	2973
5.896.1 Detailed Description	2974
5.896.2 Member Function Documentation	2975
5.897 __gnu_cxx::slist< _Tp, _Alloc > Class Template Reference	2976
5.897.1 Detailed Description	2978
5.898 std::experimental::filesystem::v1::space_info Struct Reference	2978
5.898.1 Detailed Description	2978
5.899 std::filesystem::space_info Struct Reference	2978
5.899.1 Detailed Description	2979
5.900 __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference	2979
5.900.1 Detailed Description	2981
5.900.2 Member Function Documentation	2982
5.901 __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference	2982
5.901.1 Detailed Description	2983
5.902 __gnu_pbds::splay_tree_tag Struct Reference	2983
5.902.1 Detailed Description	2984
5.903 std::stack< _Tp, _Sequence > Class Template Reference	2984
5.903.1 Detailed Description	2985
5.903.2 Constructor & Destructor Documentation	2986
5.903.3 Member Function Documentation	2986
5.904 __gnu_cxx::stdio_filebuf< _CharT, _Traits > Class Template Reference	2987
5.904.1 Detailed Description	2990
5.904.2 Constructor & Destructor Documentation	2990
5.904.3 Member Function Documentation	2991
5.904.4 Member Data Documentation	3006
5.905 __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits > Class Template Reference	3009
5.905.1 Detailed Description	3011
5.905.2 Member Function Documentation	3011
5.905.3 Member Data Documentation	3024
5.906 std::chrono::steady_clock Struct Reference	3025
5.906.1 Detailed Description	3026
5.907 __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash > Struct Template Reference	3026

5.907.1 Detailed Description	3026
5.908 <code>__gnu_pbds::detail::stored_data&lt;_Tv, _Th, false &gt;</code> Struct Template Reference	3027
5.908.1 Detailed Description	3027
5.909 <code>__gnu_pbds::detail::stored_hash&lt;_Th &gt;</code> Struct Template Reference	3027
5.909.1 Detailed Description	3028
5.910 <code>__gnu_pbds::detail::stored_value&lt;_Tv &gt;</code> Struct Template Reference	3028
5.910.1 Detailed Description	3029
5.911 <code>__gnu_pbds::string_tag</code> Struct Reference	3029
5.911.1 Detailed Description	3030
5.912 <code>std::student_t_distribution&lt;_RealType &gt;</code> Class Template Reference	3030
5.912.1 Detailed Description	3031
5.912.2 Member Typedef Documentation	3031
5.912.3 Member Function Documentation	3031
5.912.4 Friends And Related Symbol Documentation	3032
5.913 <code>std::sub_match&lt;_Bilter &gt;</code> Class Template Reference	3033
5.913.1 Detailed Description	3034
5.913.2 Member Function Documentation	3035
5.914 <code>std::ranges::subrange&lt;_It, _Sent, _Kind &gt;</code> Class Template Reference	3036
5.914.1 Detailed Description	3038
5.915 <code>std::subtract_with_carry_engine&lt;_UIntType, __w, __s, __r &gt;</code> Class Template Reference	3038
5.915.1 Detailed Description	3039
5.915.2 Member Typedef Documentation	3039
5.915.3 Constructor & Destructor Documentation	3039
5.915.4 Member Function Documentation	3040
5.915.5 Friends And Related Symbol Documentation	3041
5.916 <code>__gnu_cxx::subtractive_rng</code> Class Reference	3042
5.916.1 Detailed Description	3043
5.916.2 Member Typedef Documentation	3043
5.916.3 Constructor & Destructor Documentation	3043
5.916.4 Member Function Documentation	3043
5.917 <code>__gnu_pbds::detail::synth_access_traits&lt;Type_Traits, Set, _ATraits &gt;</code> Struct Template Reference	3043
5.917.1 Detailed Description	3044
5.918 <code>std::chrono::system_clock</code> Struct Reference	3044
5.918.1 Detailed Description	3044
5.919 <code>std::system_error</code> Class Reference	3045
5.919.1 Detailed Description	3045
5.919.2 Member Function Documentation	3045
5.920 <code>std::chrono::tai_clock</code> Class Reference	3045
5.920.1 Detailed Description	3046

5.921	<a href="#">__gnu_cxx::temporary_buffer&lt;_ForwardIterator, _Tp&gt; Struct Template Reference</a>	3046
5.921.1	Detailed Description	3047
5.921.2	Constructor & Destructor Documentation	3047
5.921.3	Member Function Documentation	3048
5.922	<a href="#">__gnu_pbds::detail::thin_heap&lt;Value_Type, Cmp_Fn, _Alloc&gt; Class Template Reference</a>	3048
5.922.1	Detailed Description	3050
5.923	<a href="#">__gnu_pbds::thin_heap_tag Struct Reference</a>	3050
5.923.1	Detailed Description	3051
5.924	<a href="#">std::thread Class Reference</a>	3051
5.924.1	Detailed Description	3051
5.924.2	Member Function Documentation	3052
5.925	<a href="#">__gnu_cxx::throw_allocator_base&lt;_Tp, _Cond&gt; Class Template Reference</a>	3052
5.925.1	Detailed Description	3053
5.926	<a href="#">__gnu_cxx::throw_allocator_limit&lt;_Tp&gt; Struct Template Reference</a>	3053
5.926.1	Detailed Description	3055
5.927	<a href="#">__gnu_cxx::throw_allocator_random&lt;_Tp&gt; Struct Template Reference</a>	3055
5.927.1	Detailed Description	3057
5.928	<a href="#">__gnu_cxx::throw_value_base&lt;_Cond&gt; Struct Template Reference</a>	3057
5.928.1	Detailed Description	3057
5.929	<a href="#">__gnu_cxx::throw_value_limit Struct Reference</a>	3058
5.929.1	Detailed Description	3059
5.930	<a href="#">__gnu_cxx::throw_value_random Struct Reference</a>	3059
5.930.1	Detailed Description	3060
5.931	<a href="#">std::time_base Class Reference</a>	3060
5.931.1	Detailed Description	3061
5.932	<a href="#">std::time_get&lt;_CharT, _InIter&gt; Class Template Reference</a>	3061
5.932.1	Detailed Description	3062
5.932.2	Member Typedef Documentation	3063
5.932.3	Constructor & Destructor Documentation	3063
5.932.4	Member Function Documentation	3063
5.932.5	Member Data Documentation	3072
5.933	<a href="#">std::time_get_byname&lt;_CharT, _InIter&gt; Class Template Reference</a>	3072
5.933.1	Detailed Description	3074
5.933.2	Member Function Documentation	3074
5.933.3	Member Data Documentation	3082
5.934	<a href="#">std::chrono::time_point&lt;_Clock, _Dur&gt; Class Template Reference</a>	3082
5.934.1	Detailed Description	3083
5.935	<a href="#">std::time_put&lt;_CharT, _OutIter&gt; Class Template Reference</a>	3083
5.935.1	Detailed Description	3085

5.935.2 Member Typedef Documentation . . . . .	3085
5.935.3 Constructor & Destructor Documentation . . . . .	3085
5.935.4 Member Function Documentation . . . . .	3085
5.935.5 Member Data Documentation . . . . .	3087
5.936 std::time_put_byname< _CharT, _OutIter > Class Template Reference . . . . .	3087
5.936.1 Detailed Description . . . . .	3089
5.936.2 Member Function Documentation . . . . .	3089
5.936.3 Member Data Documentation . . . . .	3090
5.937 std::timed_mutex Class Reference . . . . .	3091
5.937.1 Detailed Description . . . . .	3091
5.938 std::to_chars_result Struct Reference . . . . .	3091
5.938.1 Detailed Description . . . . .	3091
5.939 std::chrono::treat_as_floating_point< _Rep > Struct Template Reference . . . . .	3092
5.939.1 Detailed Description . . . . .	3092
5.940 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc > Class Template Reference . . . . .	3092
5.940.1 Detailed Description . . . . .	3093
5.940.2 Member Typedef Documentation . . . . .	3094
5.940.3 Constructor & Destructor Documentation . . . . .	3094
5.941 __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp > Struct Template Reference . . . . .	3095
5.941.1 Detailed Description . . . . .	3095
5.942 __gnu_pbds::detail::tree_metadata_helper< Node_Update, false > Struct Template Reference . . . . .	3095
5.942.1 Detailed Description . . . . .	3095
5.943 __gnu_pbds::detail::tree_metadata_helper< Node_Update, true > Struct Template Reference . . . . .	3095
5.943.1 Detailed Description . . . . .	3095
5.944 __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference . . . . .	3096
5.944.1 Detailed Description . . . . .	3096
5.945 __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc > Class Template Reference . . . . .	3096
5.945.1 Detailed Description . . . . .	3097
5.945.2 Member Function Documentation . . . . .	3097
5.946 __gnu_pbds::tree_tag Struct Reference . . . . .	3098
5.946.1 Detailed Description . . . . .	3098
5.947 __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference . . . . .	3098
5.947.1 Detailed Description . . . . .	3098
5.948 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference . . . . .	3099
5.948.1 Detailed Description . . . . .	3099
5.948.2 Member Typedef Documentation . . . . .	3099



5.949	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc &gt; Struct Template Reference</a>	3099
5.949.1	<a href="#">Detailed Description</a>	3100
5.949.2	<a href="#">Member Typedef Documentation</a>	3100
5.950	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc &gt; Struct Template Reference</a>	3100
5.950.1	<a href="#">Detailed Description</a>	3101
5.950.2	<a href="#">Member Typedef Documentation</a>	3102
5.951	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc &gt; Struct Template Reference</a>	3102
5.951.1	<a href="#">Detailed Description</a>	3102
5.951.2	<a href="#">Member Typedef Documentation</a>	3102
5.952	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc &gt; Struct Template Reference</a>	3102
5.952.1	<a href="#">Detailed Description</a>	3103
5.952.2	<a href="#">Member Typedef Documentation</a>	3104
5.953	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc &gt; Struct Template Reference</a>	3104
5.953.1	<a href="#">Detailed Description</a>	3105
5.953.2	<a href="#">Member Typedef Documentation</a>	3105
5.954	<a href="#">__gnu_pbds::trie&lt; Key, Mapped, _ATraits, Tag, Node_Update, _Alloc &gt; Class Template Reference</a>	3105
5.954.1	<a href="#">Detailed Description</a>	3106
5.954.2	<a href="#">Member Typedef Documentation</a>	3107
5.954.3	<a href="#">Constructor &amp; Destructor Documentation</a>	3107
5.955	<a href="#">__gnu_pbds::detail::trie_metadata_helper&lt; Node_Update, _BTp &gt; Struct Template Reference</a>	3108
5.955.1	<a href="#">Detailed Description</a>	3108
5.956	<a href="#">__gnu_pbds::detail::trie_metadata_helper&lt; Node_Update, false &gt; Struct Template Reference</a>	3108
5.956.1	<a href="#">Detailed Description</a>	3108
5.957	<a href="#">__gnu_pbds::detail::trie_metadata_helper&lt; Node_Update, true &gt; Struct Template Reference</a>	3108
5.957.1	<a href="#">Detailed Description</a>	3109
5.958	<a href="#">__gnu_pbds::detail::trie_node_metadata_dispatch&lt; Key, Data, Cmp_Fn, Node_Update, _Alloc &gt; Struct Template Reference</a>	3109
5.958.1	<a href="#">Detailed Description</a>	3109
5.959	<a href="#">__gnu_pbds::trie_order_statistics_node_update&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt; Class Template Reference</a>	3109
5.959.1	<a href="#">Detailed Description</a>	3111
5.959.2	<a href="#">Member Function Documentation</a>	3111
5.960	<a href="#">__gnu_pbds::detail::trie_policy_base&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt; Class Template Reference</a>	3112
5.960.1	<a href="#">Detailed Description</a>	3113
5.960.2	<a href="#">Member Function Documentation</a>	3113

5.961	<a href="#">__gnu_pbds::trie_prefix_search_node_update&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt; Class Template Reference</a>	3113
5.961.1	<a href="#">Detailed Description</a>	3115
5.961.2	<a href="#">Member Typedef Documentation</a>	3115
5.961.3	<a href="#">Member Function Documentation</a>	3115
5.962	<a href="#">__gnu_pbds::trie_string_access_traits&lt; String, Min_E_Val, Max_E_Val, Reverse, _Alloc &gt; Struct Template Reference</a>	3116
5.962.1	<a href="#">Detailed Description</a>	3117
5.962.2	<a href="#">Member Typedef Documentation</a>	3117
5.962.3	<a href="#">Member Function Documentation</a>	3117
5.963	<a href="#">__gnu_pbds::trie_tag Struct Reference</a>	3118
5.963.1	<a href="#">Detailed Description</a>	3119
5.964	<a href="#">__gnu_pbds::detail::trie_traits&lt; Key, Data, _ATraits, Node_Update, Tag, _Alloc &gt; Struct Template Reference</a>	3119
5.964.1	<a href="#">Detailed Description</a>	3119
5.965	<a href="#">__gnu_pbds::detail::trie_traits&lt; Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc &gt; Struct Template Reference</a>	3120
5.965.1	<a href="#">Detailed Description</a>	3120
5.965.2	<a href="#">Member Typedef Documentation</a>	3120
5.966	<a href="#">__gnu_pbds::detail::trie_traits&lt; Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc &gt; Struct Template Reference</a>	3121
5.966.1	<a href="#">Detailed Description</a>	3121
5.966.2	<a href="#">Member Typedef Documentation</a>	3121
5.967	<a href="#">__gnu_pbds::trivial_iterator_tag Struct Reference</a>	3122
5.967.1	<a href="#">Detailed Description</a>	3122
5.968	<a href="#">std::try_to_lock_t Struct Reference</a>	3122
5.968.1	<a href="#">Detailed Description</a>	3122
5.969	<a href="#">std::tuple&lt; _Elements &gt; Class Template Reference</a>	3122
5.969.1	<a href="#">Detailed Description</a>	3124
5.970	<a href="#">std::tuple&lt; _T1, _T2 &gt; Class Template Reference</a>	3124
5.970.1	<a href="#">Detailed Description</a>	3126
5.971	<a href="#">std::tuple_element&lt; __i, _Tp &gt; Struct Template Reference</a>	3126
5.971.1	<a href="#">Detailed Description</a>	3126
5.972	<a href="#">std::tuple_element&lt; 0, pair&lt; _Tp1, _Tp2 &gt; &gt; Struct Template Reference</a>	3127
5.972.1	<a href="#">Detailed Description</a>	3127
5.973	<a href="#">std::tuple_element&lt; 1, pair&lt; _Tp1, _Tp2 &gt; &gt; Struct Template Reference</a>	3127
5.973.1	<a href="#">Detailed Description</a>	3127
5.974	<a href="#">std::tuple_element&lt; __i, tuple&lt; _Types... &gt; &gt; Struct Template Reference</a>	3127
5.974.1	<a href="#">Detailed Description</a>	3127
5.975	<a href="#">std::tuple_element&lt; _Ind, array&lt; _Tp, _Nm &gt; &gt; Struct Template Reference</a>	3127

5.975.1 Detailed Description	3128
5.976 <code>std::tuple_size&lt; _Tp &gt;</code> Struct Template Reference	3128
5.976.1 Detailed Description	3128
5.977 <code>std::tuple_size&lt; array&lt; _Tp, _Nm &gt; &gt;</code> Struct Template Reference	3128
5.977.1 Detailed Description	3129
5.978 <code>std::tuple_size&lt; pair&lt; _Tp1, _Tp2 &gt; &gt;</code> Struct Template Reference	3129
5.978.1 Detailed Description	3130
5.979 <code>std::tuple_size&lt; tuple&lt; _Elements... &gt; &gt;</code> Struct Template Reference	3130
5.979.1 Detailed Description	3130
5.980 <code>__gnu_pbds::detail::entry_cmp&lt; _VTp, Cmp_Fn, _Alloc, false &gt;::type</code> Struct Reference	3131
5.980.1 Detailed Description	3131
5.981 <code>std::type_index</code> Struct Reference	3131
5.981.1 Detailed Description	3131
5.982 <code>__gnu_debug::type_info</code> Class Reference	3131
5.982.1 Detailed Description	3132
5.982.2 Constructor & Destructor Documentation	3132
5.982.3 Member Function Documentation	3132
5.983 <code>std::type_info</code> Class Reference	3132
5.983.1 Detailed Description	3133
5.983.2 Constructor & Destructor Documentation	3133
5.983.3 Member Function Documentation	3133
5.984 <code>__gnu_pbds::detail::types_traits&lt; Key, Mapped, _Alloc, Store_Hash &gt;</code> Struct Template Reference	3133
5.984.1 Detailed Description	3135
5.985 <code>std::chrono::tzdb_list</code> Class Reference	3135
5.985.1 Detailed Description	3135
5.985.2 Member Function Documentation	3136
5.986 <code>__gnu_cxx::unary_compose&lt; _Operation1, _Operation2 &gt;</code> Class Template Reference	3136
5.986.1 Detailed Description	3137
5.986.2 Member Typedef Documentation	3137
5.987 <code>std::unary_function&lt; _Arg, _Result &gt;</code> Struct Template Reference	3138
5.987.1 Detailed Description	3138
5.987.2 Member Typedef Documentation	3138
5.988 <code>std::unary_negate&lt; _Predicate &gt;</code> Class Template Reference	3139
5.988.1 Detailed Description	3139
5.988.2 Member Typedef Documentation	3139
5.989 <code>__gnu_parallel::unbalanced_tag</code> Struct Reference	3140
5.989.1 Detailed Description	3140
5.989.2 Member Function Documentation	3140
5.990 <code>std::underflow_error</code> Class Reference	3141

5.990.1 Detailed Description	3141
5.990.2 Member Function Documentation	3141
5.991 std::underlying_type< _Tp > Struct Template Reference	3142
5.991.1 Detailed Description	3142
5.992 std::uniform_int_distribution< _IntType > Class Template Reference	3142
5.992.1 Detailed Description	3143
5.992.2 Member Typedef Documentation	3143
5.992.3 Constructor & Destructor Documentation	3143
5.992.4 Member Function Documentation	3143
5.992.5 Friends And Related Symbol Documentation	3144
5.993 std::uniform_real_distribution< _RealType > Class Template Reference	3144
5.993.1 Detailed Description	3145
5.993.2 Member Typedef Documentation	3145
5.993.3 Constructor & Destructor Documentation	3145
5.993.4 Member Function Documentation	3146
5.993.5 Friends And Related Symbol Documentation	3147
5.994 std::unique_lock< _Mutex > Class Template Reference	3147
5.994.1 Detailed Description	3148
5.994.2 Friends And Related Symbol Documentation	3148
5.995 std::unique_ptr< _Tp, _Dp > Class Template Reference	3148
5.995.1 Detailed Description	3149
5.995.2 Constructor & Destructor Documentation	3150
5.995.3 Member Function Documentation	3151
5.996 std::unique_ptr< _Tp[], _Dp > Class Template Reference	3153
5.996.1 Detailed Description	3155
5.996.2 Constructor & Destructor Documentation	3156
5.996.3 Member Function Documentation	3159
5.996.4 Friends And Related Symbol Documentation	3162
5.997 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3163
5.997.1 Detailed Description	3166
5.998 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3166
5.998.1 Detailed Description	3169
5.998.2 Member Typedef Documentation	3170
5.998.3 Constructor & Destructor Documentation	3172
5.998.4 Member Function Documentation	3174
5.999 std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3194
5.999.1 Detailed Description	3197
5.1000 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3197
5.1000.1 Detailed Description	3200

5.1000.2 Member Typedef Documentation . . . . .	3200
5.1000.3 Constructor & Destructor Documentation . . . . .	3203
5.1000.4 Member Function Documentation . . . . .	3205
5.1001 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	3223
5.1001.1 Detailed Description . . . . .	3225
5.1002 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	3225
5.1002.1 Detailed Description . . . . .	3228
5.1002.2 Member Typedef Documentation . . . . .	3229
5.1002.3 Constructor & Destructor Documentation . . . . .	3231
5.1002.4 Member Function Documentation . . . . .	3233
5.1003 std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	3249
5.1003.1 Detailed Description . . . . .	3252
5.1004 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	3252
5.1004.1 Detailed Description . . . . .	3255
5.1004.2 Member Typedef Documentation . . . . .	3255
5.1004.3 Constructor & Destructor Documentation . . . . .	3257
5.1004.4 Member Function Documentation . . . . .	3259
5.1005 std::pmr::unsynchronized_pool_resource Class Reference . . . . .	3277
5.1005.1 Detailed Description . . . . .	3278
5.1005.2 Member Function Documentation . . . . .	3278
5.1006 std::uses_allocator< typename, typename > Struct Template Reference . . . . .	3279
5.1006.1 Detailed Description . . . . .	3279
5.1007 std::uses_allocator< tuple< _Types... >, _Alloc > Struct Template Reference . . . . .	3279
5.1007.1 Detailed Description . . . . .	3279
5.1008 std::chrono::utc_clock Class Reference . . . . .	3279
5.1008.1 Detailed Description . . . . .	3279
5.1009 std::valarray< _Tp > Class Template Reference . . . . .	3280
5.1009.1 Detailed Description . . . . .	3282
5.1009.2 Constructor & Destructor Documentation . . . . .	3282
5.1009.3 Member Function Documentation . . . . .	3282
5.1010 std::__debug::vector< _Tp, _Allocator > Class Template Reference . . . . .	3285
5.1010.1 Detailed Description . . . . .	3287
5.1010.2 Constructor & Destructor Documentation . . . . .	3288
5.1011 std::vector< _Tp, _Alloc > Class Template Reference . . . . .	3288
5.1011.1 Detailed Description . . . . .	3291
5.1011.2 Constructor & Destructor Documentation . . . . .	3292
5.1011.3 Member Function Documentation . . . . .	3294
5.1012 std::vector< bool, _Alloc > Class Template Reference . . . . .	3307
5.1012.1 Detailed Description . . . . .	3312

5.1012.2 Constructor & Destructor Documentation . . . . .	3312
5.1012.3 Member Function Documentation . . . . .	3315
5.1013 std::ranges::view_interface< _Derived > Class Template Reference . . . . .	3327
5.1013.1 Detailed Description . . . . .	3327
5.1014 std::wbuffer_convert< _Codecvt, _Elem, _Tr > Class Template Reference . . . . .	3328
5.1014.1 Detailed Description . . . . .	3330
5.1014.2 Member Typedef Documentation . . . . .	3330
5.1014.3 Constructor & Destructor Documentation . . . . .	3330
5.1014.4 Member Function Documentation . . . . .	3331
5.1014.5 Member Data Documentation . . . . .	3343
5.1015 std::weak_ptr< _Tp > Class Template Reference . . . . .	3344
5.1015.1 Detailed Description . . . . .	3345
5.1016 std::weibull_distribution< _RealType > Class Template Reference . . . . .	3345
5.1016.1 Detailed Description . . . . .	3346
5.1016.2 Member Typedef Documentation . . . . .	3346
5.1016.3 Member Function Documentation . . . . .	3346
5.1016.4 Friends And Related Symbol Documentation . . . . .	3347
5.1017 std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc > Class Template Reference . . . . .	3347
5.1017.1 Detailed Description . . . . .	3348
5.1017.2 Constructor & Destructor Documentation . . . . .	3348
5.1017.3 Member Function Documentation . . . . .	3349
<b>6 File Documentation</b>	<b>3351</b>
6.1 algorithm File Reference . . . . .	3351
6.1.1 Detailed Description . . . . .	3351
6.2 experimental/algorithm File Reference . . . . .	3351
6.2.1 Detailed Description . . . . .	3352
6.3 ext/algorithm File Reference . . . . .	3352
6.3.1 Detailed Description . . . . .	3353
6.4 parallel/algorithm File Reference . . . . .	3353
6.4.1 Detailed Description . . . . .	3353
6.5 any File Reference . . . . .	3353
6.5.1 Detailed Description . . . . .	3353
6.6 experimental/any File Reference . . . . .	3354
6.6.1 Detailed Description . . . . .	3354
6.7 array File Reference . . . . .	3354
6.7.1 Detailed Description . . . . .	3355
6.8 experimental/array File Reference . . . . .	3355
6.8.1 Detailed Description . . . . .	3356

6.9 atomic File Reference . . . . .	3356
6.9.1 Detailed Description . . . . .	3359
6.10 auto_ptr.h File Reference . . . . .	3360
6.10.1 Detailed Description . . . . .	3360
6.11 backward_warning.h File Reference . . . . .	3360
6.11.1 Detailed Description . . . . .	3360
6.12 hash_fun.h File Reference . . . . .	3360
6.12.1 Detailed Description . . . . .	3360
6.13 hash_map File Reference . . . . .	3360
6.13.1 Detailed Description . . . . .	3361
6.14 hash_set File Reference . . . . .	3361
6.14.1 Detailed Description . . . . .	3362
6.15 stringstream File Reference . . . . .	3362
6.15.1 Detailed Description . . . . .	3362
6.16 barrier File Reference . . . . .	3362
6.16.1 Detailed Description . . . . .	3362
6.17 bit File Reference . . . . .	3362
6.17.1 Detailed Description . . . . .	3363
6.18 bits/algorithmfwd.h File Reference . . . . .	3363
6.18.1 Detailed Description . . . . .	3368
6.19 parallel/algorithmfwd.h File Reference . . . . .	3368
6.19.1 Detailed Description . . . . .	3376
6.20 align.h File Reference . . . . .	3376
6.20.1 Detailed Description . . . . .	3376
6.21 bits/alloc_traits.h File Reference . . . . .	3376
6.21.1 Detailed Description . . . . .	3377
6.22 ext/alloc_traits.h File Reference . . . . .	3377
6.22.1 Detailed Description . . . . .	3377
6.23 allocated_ptr.h File Reference . . . . .	3377
6.23.1 Detailed Description . . . . .	3377
6.24 allocator.h File Reference . . . . .	3377
6.24.1 Detailed Description . . . . .	3377
6.25 atomic_base.h File Reference . . . . .	3377
6.25.1 Detailed Description . . . . .	3378
6.26 atomic_futex.h File Reference . . . . .	3378
6.26.1 Detailed Description . . . . .	3378
6.27 atomic_lockfree_defines.h File Reference . . . . .	3378
6.27.1 Detailed Description . . . . .	3379
6.28 atomic_timed_wait.h File Reference . . . . .	3379

6.28.1 Detailed Description	3379
6.29 atomic_wait.h File Reference	3379
6.29.1 Detailed Description	3379
6.30 basic_ios.h File Reference	3379
6.30.1 Detailed Description	3379
6.31 basic_ios.tcc File Reference	3379
6.31.1 Detailed Description	3380
6.32 basic_string.h File Reference	3380
6.32.1 Detailed Description	3382
6.33 basic_string.tcc File Reference	3382
6.33.1 Detailed Description	3383
6.34 backward/binders.h File Reference	3383
6.34.1 Detailed Description	3383
6.35 boost_concept_check.h File Reference	3383
6.35.1 Detailed Description	3384
6.36 c++0x_warning.h File Reference	3384
6.36.1 Detailed Description	3384
6.37 char_traits.h File Reference	3384
6.37.1 Detailed Description	3384
6.38 charconv.h File Reference	3385
6.38.1 Detailed Description	3385
6.39 chrono.h File Reference	3385
6.39.1 Detailed Description	3388
6.39.2 Function Documentation	3389
6.40 chrono_io.h File Reference	3389
6.40.1 Detailed Description	3392
6.41 codecvt.h File Reference	3393
6.41.1 Detailed Description	3393
6.42 concept_check.h File Reference	3393
6.42.1 Detailed Description	3393
6.43 cow_string.h File Reference	3393
6.43.1 Detailed Description	3394
6.44 cpp_type_traits.h File Reference	3394
6.44.1 Detailed Description	3394
6.45 cpyfunc_impl.h File Reference	3394
6.45.1 Detailed Description	3395
6.46 cxxabi_forced.h File Reference	3395
6.46.1 Detailed Description	3395
6.47 cxxabi_init_exception.h File Reference	3395



6.47.1 Detailed Description . . . . .	3395
6.48 deque.tcc File Reference . . . . .	3395
6.48.1 Detailed Description . . . . .	3397
6.49 enable_special_members.h File Reference . . . . .	3397
6.49.1 Detailed Description . . . . .	3397
6.50 erase_if.h File Reference . . . . .	3397
6.50.1 Detailed Description . . . . .	3397
6.51 exception.h File Reference . . . . .	3397
6.51.1 Detailed Description . . . . .	3398
6.52 exception_defines.h File Reference . . . . .	3398
6.52.1 Detailed Description . . . . .	3398
6.53 exception_ptr.h File Reference . . . . .	3398
6.53.1 Detailed Description . . . . .	3398
6.53.2 Function Documentation . . . . .	3398
6.54 format_fwd.h File Reference . . . . .	3398
6.54.1 Detailed Description . . . . .	3398
6.55 forward_list.h File Reference . . . . .	3399
6.55.1 Detailed Description . . . . .	3399
6.56 forward_list.tcc File Reference . . . . .	3399
6.56.1 Detailed Description . . . . .	3400
6.57 bits/fs_dir.h File Reference . . . . .	3400
6.57.1 Detailed Description . . . . .	3400
6.58 experimental/bits/fs_dir.h File Reference . . . . .	3400
6.58.1 Detailed Description . . . . .	3401
6.59 bits/fs_fwd.h File Reference . . . . .	3401
6.59.1 Detailed Description . . . . .	3403
6.60 experimental/bits/fs_fwd.h File Reference . . . . .	3403
6.60.1 Detailed Description . . . . .	3404
6.61 bits/fs_ops.h File Reference . . . . .	3404
6.61.1 Detailed Description . . . . .	3406
6.62 experimental/bits/fs_ops.h File Reference . . . . .	3407
6.62.1 Detailed Description . . . . .	3409
6.63 bits/fs_path.h File Reference . . . . .	3409
6.63.1 Detailed Description . . . . .	3409
6.64 experimental/bits/fs_path.h File Reference . . . . .	3409
6.64.1 Detailed Description . . . . .	3410
6.65 fstream.tcc File Reference . . . . .	3410
6.65.1 Detailed Description . . . . .	3410
6.66 funcref_impl.h File Reference . . . . .	3410

---

6.66.1 Detailed Description . . . . .	3410
6.67 funtexcept.h File Reference . . . . .	3410
6.67.1 Detailed Description . . . . .	3411
6.68 functional_hash.h File Reference . . . . .	3411
6.68.1 Detailed Description . . . . .	3412
6.69 funcwrap.h File Reference . . . . .	3412
6.69.1 Detailed Description . . . . .	3412
6.70 gslice.h File Reference . . . . .	3412
6.70.1 Detailed Description . . . . .	3412
6.71 gslice_array.h File Reference . . . . .	3412
6.71.1 Detailed Description . . . . .	3412
6.72 hash_bytes.h File Reference . . . . .	3412
6.72.1 Detailed Description . . . . .	3413
6.73 backward/hashtable.h File Reference . . . . .	3413
6.73.1 Detailed Description . . . . .	3413
6.74 bits/hashtable.h File Reference . . . . .	3413
6.74.1 Detailed Description . . . . .	3413
6.75 hashtable_policy.h File Reference . . . . .	3414
6.75.1 Detailed Description . . . . .	3414
6.76 indirect.h File Reference . . . . .	3414
6.76.1 Detailed Description . . . . .	3414
6.77 indirect_array.h File Reference . . . . .	3414
6.77.1 Detailed Description . . . . .	3414
6.78 invoke.h File Reference . . . . .	3414
6.78.1 Detailed Description . . . . .	3415
6.79 ios_base.h File Reference . . . . .	3415
6.79.1 Detailed Description . . . . .	3416
6.80 istream.tcc File Reference . . . . .	3416
6.80.1 Detailed Description . . . . .	3417
6.81 iterator_concepts.h File Reference . . . . .	3417
6.81.1 Detailed Description . . . . .	3417
6.82 list.tcc File Reference . . . . .	3417
6.82.1 Detailed Description . . . . .	3417
6.83 locale_classes.h File Reference . . . . .	3418
6.83.1 Detailed Description . . . . .	3418
6.84 locale_classes.tcc File Reference . . . . .	3418
6.84.1 Detailed Description . . . . .	3418
6.85 locale_conv.h File Reference . . . . .	3418
6.85.1 Detailed Description . . . . .	3419

6.86 locale_facets.h File Reference . . . . .	3419
6.86.1 Detailed Description . . . . .	3421
6.87 locale_facets.tcc File Reference . . . . .	3421
6.87.1 Detailed Description . . . . .	3421
6.88 locale_facets_nonio.h File Reference . . . . .	3421
6.88.1 Detailed Description . . . . .	3422
6.89 locale_facets_nonio.tcc File Reference . . . . .	3422
6.89.1 Detailed Description . . . . .	3422
6.90 localefwd.h File Reference . . . . .	3423
6.90.1 Detailed Description . . . . .	3423
6.91 mask_array.h File Reference . . . . .	3423
6.91.1 Detailed Description . . . . .	3424
6.92 max_size_type.h File Reference . . . . .	3424
6.92.1 Detailed Description . . . . .	3424
6.93 memory_resource.h File Reference . . . . .	3424
6.93.1 Detailed Description . . . . .	3424
6.94 memoryfwd.h File Reference . . . . .	3424
6.94.1 Detailed Description . . . . .	3424
6.95 mofunc_impl.h File Reference . . . . .	3424
6.95.1 Detailed Description . . . . .	3425
6.96 monostate.h File Reference . . . . .	3425
6.96.1 Detailed Description . . . . .	3425
6.97 move.h File Reference . . . . .	3425
6.97.1 Detailed Description . . . . .	3426
6.98 nested_exception.h File Reference . . . . .	3426
6.98.1 Detailed Description . . . . .	3426
6.99 bits/new_allocator.h File Reference . . . . .	3426
6.99.1 Detailed Description . . . . .	3427
6.100 ext/new_allocator.h File Reference . . . . .	3427
6.100.1 Detailed Description . . . . .	3427
6.101 node_handle.h File Reference . . . . .	3427
6.101.1 Detailed Description . . . . .	3427
6.102 ostream.h File Reference . . . . .	3427
6.102.1 Detailed Description . . . . .	3428
6.103 ostream.tcc File Reference . . . . .	3429
6.103.1 Detailed Description . . . . .	3429
6.104 ostream_insert.h File Reference . . . . .	3429
6.104.1 Detailed Description . . . . .	3429
6.105 out_ptr.h File Reference . . . . .	3429

---

6.105.1 Detailed Description . . . . .	3429
6.106 parse_numbers.h File Reference . . . . .	3429
6.106.1 Detailed Description . . . . .	3429
6.107 postypes.h File Reference . . . . .	3430
6.107.1 Detailed Description . . . . .	3430
6.108 predefined_ops.h File Reference . . . . .	3430
6.108.1 Detailed Description . . . . .	3431
6.109 ptr_traits.h File Reference . . . . .	3431
6.109.1 Detailed Description . . . . .	3431
6.110 quoted_string.h File Reference . . . . .	3432
6.110.1 Detailed Description . . . . .	3432
6.111 random.h File Reference . . . . .	3432
6.111.1 Detailed Description . . . . .	3435
6.112 bits/random.tcc File Reference . . . . .	3435
6.112.1 Detailed Description . . . . .	3438
6.113 ext/random.tcc File Reference . . . . .	3438
6.113.1 Detailed Description . . . . .	3440
6.114 range_access.h File Reference . . . . .	3440
6.114.1 Detailed Description . . . . .	3442
6.115 ranges_algo.h File Reference . . . . .	3442
6.115.1 Detailed Description . . . . .	3446
6.116 ranges_algobase.h File Reference . . . . .	3447
6.116.1 Detailed Description . . . . .	3447
6.117 ranges_base.h File Reference . . . . .	3447
6.117.1 Detailed Description . . . . .	3447
6.118 ranges_cmp.h File Reference . . . . .	3447
6.118.1 Detailed Description . . . . .	3447
6.119 ranges_uninitialized.h File Reference . . . . .	3447
6.119.1 Detailed Description . . . . .	3447
6.120 ranges_util.h File Reference . . . . .	3447
6.120.1 Detailed Description . . . . .	3448
6.121 refwrap.h File Reference . . . . .	3449
6.121.1 Detailed Description . . . . .	3449
6.122 regex.h File Reference . . . . .	3449
6.122.1 Detailed Description . . . . .	3451
6.123 regex.tcc File Reference . . . . .	3451
6.123.1 Detailed Description . . . . .	3451
6.124 regex_automaton.h File Reference . . . . .	3452
6.124.1 Detailed Description . . . . .	3452

6.125 regex_automaton.tcc File Reference . . . . .	3452
6.125.1 Detailed Description . . . . .	3452
6.126 regex_compiler.h File Reference . . . . .	3452
6.126.1 Detailed Description . . . . .	3453
6.127 regex_compiler.tcc File Reference . . . . .	3453
6.127.1 Detailed Description . . . . .	3453
6.128 regex_constants.h File Reference . . . . .	3453
6.128.1 Detailed Description . . . . .	3454
6.129 regex_error.h File Reference . . . . .	3455
6.129.1 Detailed Description . . . . .	3455
6.130 regex_executor.h File Reference . . . . .	3455
6.130.1 Detailed Description . . . . .	3455
6.131 regex_executor.tcc File Reference . . . . .	3456
6.131.1 Detailed Description . . . . .	3456
6.132 regex_scanner.h File Reference . . . . .	3456
6.132.1 Detailed Description . . . . .	3456
6.133 regex_scanner.tcc File Reference . . . . .	3456
6.133.1 Detailed Description . . . . .	3456
6.134 requires_hosted.h File Reference . . . . .	3456
6.134.1 Detailed Description . . . . .	3456
6.135 sat_arith.h File Reference . . . . .	3456
6.135.1 Detailed Description . . . . .	3456
6.136 semaphore_base.h File Reference . . . . .	3457
6.136.1 Detailed Description . . . . .	3457
6.137 bits/shared_ptr.h File Reference . . . . .	3457
6.137.1 Detailed Description . . . . .	3457
6.138 experimental/bits/shared_ptr.h File Reference . . . . .	3457
6.138.1 Detailed Description . . . . .	3459
6.139 shared_ptr_atomic.h File Reference . . . . .	3459
6.139.1 Detailed Description . . . . .	3461
6.140 shared_ptr_base.h File Reference . . . . .	3461
6.140.1 Detailed Description . . . . .	3462
6.141 slice_array.h File Reference . . . . .	3462
6.141.1 Detailed Description . . . . .	3462
6.142 specfun.h File Reference . . . . .	3463
6.142.1 Detailed Description . . . . .	3465
6.143 sstream.tcc File Reference . . . . .	3465
6.143.1 Detailed Description . . . . .	3465
6.144 std_abs.h File Reference . . . . .	3465

---

6.144.1 Detailed Description . . . . .	3465
6.145 std_function.h File Reference . . . . .	3465
6.145.1 Detailed Description . . . . .	3466
6.146 std_mutex.h File Reference . . . . .	3466
6.146.1 Detailed Description . . . . .	3466
6.147 std_thread.h File Reference . . . . .	3467
6.147.1 Detailed Description . . . . .	3467
6.148 stl_algo.h File Reference . . . . .	3467
6.148.1 Detailed Description . . . . .	3476
6.149 stl_algobase.h File Reference . . . . .	3476
6.149.1 Detailed Description . . . . .	3481
6.150 stl_bvector.h File Reference . . . . .	3481
6.150.1 Detailed Description . . . . .	3482
6.151 stl_construct.h File Reference . . . . .	3482
6.151.1 Detailed Description . . . . .	3483
6.152 stl_deque.h File Reference . . . . .	3483
6.152.1 Detailed Description . . . . .	3483
6.152.2 Macro Definition Documentation . . . . .	3483
6.153 stl_function.h File Reference . . . . .	3484
6.153.1 Detailed Description . . . . .	3485
6.154 stl_heap.h File Reference . . . . .	3485
6.154.1 Detailed Description . . . . .	3487
6.155 bits/stl_iterator.h File Reference . . . . .	3487
6.155.1 Detailed Description . . . . .	3489
6.156 debug/stl_iterator.h File Reference . . . . .	3489
6.156.1 Detailed Description . . . . .	3490
6.157 stl_iterator_base_funcs.h File Reference . . . . .	3490
6.157.1 Detailed Description . . . . .	3491
6.158 stl_iterator_base_types.h File Reference . . . . .	3491
6.158.1 Detailed Description . . . . .	3492
6.159 stl_list.h File Reference . . . . .	3492
6.159.1 Detailed Description . . . . .	3493
6.160 stl_map.h File Reference . . . . .	3493
6.160.1 Detailed Description . . . . .	3493
6.161 stl_multimap.h File Reference . . . . .	3494
6.161.1 Detailed Description . . . . .	3494
6.162 stl_multiset.h File Reference . . . . .	3495
6.162.1 Detailed Description . . . . .	3495
6.163 stl_numeric.h File Reference . . . . .	3496

6.163.1 Detailed Description . . . . .	3496
6.164 <a href="#">std_pair.h</a> File Reference . . . . .	3496
6.164.1 Detailed Description . . . . .	3497
6.165 <a href="#">std_queue.h</a> File Reference . . . . .	3497
6.165.1 Detailed Description . . . . .	3498
6.166 <a href="#">std_raw_storage_iter.h</a> File Reference . . . . .	3499
6.166.1 Detailed Description . . . . .	3499
6.167 <a href="#">std_relops.h</a> File Reference . . . . .	3499
6.167.1 Detailed Description . . . . .	3499
6.168 <a href="#">std_set.h</a> File Reference . . . . .	3499
6.168.1 Detailed Description . . . . .	3500
6.169 <a href="#">std_stack.h</a> File Reference . . . . .	3500
6.169.1 Detailed Description . . . . .	3501
6.170 <a href="#">std_tempbuf.h</a> File Reference . . . . .	3501
6.170.1 Detailed Description . . . . .	3501
6.171 <a href="#">std_tree.h</a> File Reference . . . . .	3502
6.171.1 Detailed Description . . . . .	3502
6.172 <a href="#">std_uninitialized.h</a> File Reference . . . . .	3502
6.172.1 Detailed Description . . . . .	3503
6.173 <a href="#">std_vector.h</a> File Reference . . . . .	3503
6.173.1 Detailed Description . . . . .	3503
6.174 <a href="#">stream_iterator.h</a> File Reference . . . . .	3504
6.174.1 Detailed Description . . . . .	3504
6.175 <a href="#">streambuf.tcc</a> File Reference . . . . .	3504
6.175.1 Detailed Description . . . . .	3504
6.176 <a href="#">streambuf_iterator.h</a> File Reference . . . . .	3504
6.176.1 Detailed Description . . . . .	3505
6.177 <a href="#">bits/string_view.tcc</a> File Reference . . . . .	3505
6.177.1 Detailed Description . . . . .	3505
6.178 <a href="#">experimental/bits/string_view.tcc</a> File Reference . . . . .	3505
6.178.1 Detailed Description . . . . .	3506
6.179 <a href="#">stringfwd.h</a> File Reference . . . . .	3506
6.179.1 Detailed Description . . . . .	3506
6.180 <a href="#">text_encoding-data.h</a> File Reference . . . . .	3506
6.180.1 Detailed Description . . . . .	3506
6.181 <a href="#">this_thread_sleep.h</a> File Reference . . . . .	3506
6.181.1 Detailed Description . . . . .	3506
6.182 <a href="#">unicode-data.h</a> File Reference . . . . .	3507
6.182.1 Detailed Description . . . . .	3507

6.183 unicode.h File Reference . . . . .	3507
6.183.1 Detailed Description . . . . .	3508
6.184 uniform_int_dist.h File Reference . . . . .	3508
6.184.1 Detailed Description . . . . .	3509
6.185 unique_lock.h File Reference . . . . .	3509
6.185.1 Detailed Description . . . . .	3509
6.186 unique_ptr.h File Reference . . . . .	3509
6.186.1 Detailed Description . . . . .	3510
6.187 unordered_map.h File Reference . . . . .	3510
6.187.1 Detailed Description . . . . .	3512
6.188 unordered_set.h File Reference . . . . .	3512
6.188.1 Detailed Description . . . . .	3514
6.189 uses_allocator_args.h File Reference . . . . .	3515
6.189.1 Detailed Description . . . . .	3515
6.190 utility.h File Reference . . . . .	3515
6.190.1 Detailed Description . . . . .	3516
6.191 valarray_after.h File Reference . . . . .	3516
6.191.1 Detailed Description . . . . .	3526
6.192 valarray_array.h File Reference . . . . .	3526
6.192.1 Detailed Description . . . . .	3533
6.193 valarray_array.tcc File Reference . . . . .	3534
6.193.1 Detailed Description . . . . .	3534
6.194 valarray_before.h File Reference . . . . .	3534
6.194.1 Detailed Description . . . . .	3534
6.195 vector.tcc File Reference . . . . .	3535
6.195.1 Detailed Description . . . . .	3535
6.196 version.h File Reference . . . . .	3535
6.196.1 Detailed Description . . . . .	3537
6.197 bitset File Reference . . . . .	3538
6.197.1 Detailed Description . . . . .	3538
6.198 debug/bitset File Reference . . . . .	3538
6.198.1 Detailed Description . . . . .	3539
6.199 cassert File Reference . . . . .	3539
6.199.1 Detailed Description . . . . .	3539
6.200 ccomplex File Reference . . . . .	3539
6.200.1 Detailed Description . . . . .	3539
6.201 tr1/ccomplex File Reference . . . . .	3539
6.201.1 Detailed Description . . . . .	3540
6.202 ctype File Reference . . . . .	3540



6.202.1 Detailed Description . . . . .	3540
6.203 tr1/cctype File Reference . . . . .	3540
6.203.1 Detailed Description . . . . .	3540
6.204 cerrno File Reference . . . . .	3540
6.204.1 Detailed Description . . . . .	3540
6.205 cenv File Reference . . . . .	3540
6.205.1 Detailed Description . . . . .	3540
6.206 tr1/cenv File Reference . . . . .	3541
6.206.1 Detailed Description . . . . .	3541
6.207 cfloat File Reference . . . . .	3541
6.207.1 Detailed Description . . . . .	3541
6.208 tr1/cfloat File Reference . . . . .	3541
6.208.1 Detailed Description . . . . .	3541
6.209 charconv File Reference . . . . .	3541
6.209.1 Detailed Description . . . . .	3543
6.210 chrono File Reference . . . . .	3543
6.210.1 Detailed Description . . . . .	3546
6.211 experimental/chrono File Reference . . . . .	3546
6.211.1 Detailed Description . . . . .	3546
6.212 cinttypes File Reference . . . . .	3546
6.212.1 Detailed Description . . . . .	3546
6.213 tr1/cinttypes File Reference . . . . .	3546
6.213.1 Detailed Description . . . . .	3546
6.214 ciso646 File Reference . . . . .	3546
6.214.1 Detailed Description . . . . .	3546
6.215 climits File Reference . . . . .	3547
6.215.1 Detailed Description . . . . .	3547
6.216 tr1/climits File Reference . . . . .	3547
6.216.1 Detailed Description . . . . .	3547
6.217 locale File Reference . . . . .	3547
6.217.1 Detailed Description . . . . .	3547
6.218 cmath File Reference . . . . .	3547
6.218.1 Detailed Description . . . . .	3558
6.219 ext/cmath File Reference . . . . .	3558
6.219.1 Detailed Description . . . . .	3559
6.220 tr1/cmath File Reference . . . . .	3559
6.220.1 Detailed Description . . . . .	3561
6.221 codecvt File Reference . . . . .	3561
6.221.1 Detailed Description . . . . .	3561

6.222 complex File Reference . . . . .	3562
6.222.1 Detailed Description . . . . .	3566
6.223 tr1/complex File Reference . . . . .	3566
6.223.1 Detailed Description . . . . .	3567
6.224 complex.h File Reference . . . . .	3567
6.224.1 Detailed Description . . . . .	3567
6.225 concepts File Reference . . . . .	3567
6.225.1 Detailed Description . . . . .	3567
6.226 condition_variable File Reference . . . . .	3567
6.226.1 Detailed Description . . . . .	3568
6.227 csetjmp File Reference . . . . .	3568
6.227.1 Detailed Description . . . . .	3568
6.228 csignal File Reference . . . . .	3568
6.228.1 Detailed Description . . . . .	3568
6.229 cstdalign File Reference . . . . .	3569
6.229.1 Detailed Description . . . . .	3569
6.230 cstdarg File Reference . . . . .	3569
6.230.1 Detailed Description . . . . .	3569
6.231 tr1/cstdarg File Reference . . . . .	3569
6.231.1 Detailed Description . . . . .	3569
6.232 cstdbool File Reference . . . . .	3569
6.232.1 Detailed Description . . . . .	3569
6.233 tr1/cstdbool File Reference . . . . .	3570
6.233.1 Detailed Description . . . . .	3570
6.234 cstddef File Reference . . . . .	3570
6.234.1 Detailed Description . . . . .	3570
6.235 cstdint File Reference . . . . .	3570
6.235.1 Detailed Description . . . . .	3570
6.236 tr1/cstdint File Reference . . . . .	3570
6.236.1 Detailed Description . . . . .	3570
6.237 cstdio File Reference . . . . .	3571
6.237.1 Detailed Description . . . . .	3571
6.238 tr1/cstdio File Reference . . . . .	3571
6.238.1 Detailed Description . . . . .	3571
6.239 cstdlib File Reference . . . . .	3571
6.239.1 Detailed Description . . . . .	3571
6.240 tr1/cstdlib File Reference . . . . .	3571
6.240.1 Detailed Description . . . . .	3572
6.241 cstring File Reference . . . . .	3572

6.241.1 Detailed Description . . . . .	3572
6.242 ctgmath File Reference . . . . .	3572
6.242.1 Detailed Description . . . . .	3572
6.243 tr1/ctgmath File Reference . . . . .	3572
6.243.1 Detailed Description . . . . .	3572
6.244 ctime File Reference . . . . .	3573
6.244.1 Detailed Description . . . . .	3573
6.245 tr1/ctime File Reference . . . . .	3573
6.245.1 Detailed Description . . . . .	3573
6.246 cuchar File Reference . . . . .	3573
6.246.1 Detailed Description . . . . .	3573
6.247 cwchar File Reference . . . . .	3573
6.247.1 Detailed Description . . . . .	3574
6.248 tr1/cwchar File Reference . . . . .	3574
6.248.1 Detailed Description . . . . .	3574
6.249 cwctype File Reference . . . . .	3574
6.249.1 Detailed Description . . . . .	3574
6.250 tr1/cwctype File Reference . . . . .	3574
6.250.1 Detailed Description . . . . .	3574
6.251 assertions.h File Reference . . . . .	3575
6.251.1 Detailed Description . . . . .	3575
6.252 debug.h File Reference . . . . .	3575
6.252.1 Detailed Description . . . . .	3575
6.253 formatter.h File Reference . . . . .	3576
6.253.1 Detailed Description . . . . .	3576
6.254 functions.h File Reference . . . . .	3576
6.254.1 Detailed Description . . . . .	3578
6.255 helper_functions.h File Reference . . . . .	3578
6.255.1 Detailed Description . . . . .	3580
6.256 macros.h File Reference . . . . .	3580
6.256.1 Detailed Description . . . . .	3581
6.256.2 Macro Definition Documentation . . . . .	3581
6.257 map.h File Reference . . . . .	3583
6.257.1 Detailed Description . . . . .	3584
6.258 multimap.h File Reference . . . . .	3584
6.258.1 Detailed Description . . . . .	3584
6.259 multiset.h File Reference . . . . .	3585
6.259.1 Detailed Description . . . . .	3585
6.260 safe_base.h File Reference . . . . .	3585

6.260.1 Detailed Description . . . . .	3586
6.261 safe_container.h File Reference . . . . .	3586
6.261.1 Detailed Description . . . . .	3586
6.262 safe_iterator.h File Reference . . . . .	3586
6.262.1 Detailed Description . . . . .	3587
6.263 safe_iterator.tcc File Reference . . . . .	3587
6.263.1 Detailed Description . . . . .	3588
6.264 safe_local_iterator.h File Reference . . . . .	3588
6.264.1 Detailed Description . . . . .	3588
6.265 safe_local_iterator.tcc File Reference . . . . .	3589
6.265.1 Detailed Description . . . . .	3589
6.266 safe_sequence.h File Reference . . . . .	3589
6.266.1 Detailed Description . . . . .	3589
6.267 safe_sequence.tcc File Reference . . . . .	3589
6.267.1 Detailed Description . . . . .	3589
6.268 safe_unordered_base.h File Reference . . . . .	3589
6.268.1 Detailed Description . . . . .	3590
6.269 safe_unordered_container.h File Reference . . . . .	3590
6.269.1 Detailed Description . . . . .	3590
6.270 safe_unordered_container.tcc File Reference . . . . .	3590
6.270.1 Detailed Description . . . . .	3590
6.271 set.h File Reference . . . . .	3590
6.271.1 Detailed Description . . . . .	3591
6.272 decimal File Reference . . . . .	3591
6.272.1 Detailed Description . . . . .	3600
6.273 debug/deque File Reference . . . . .	3600
6.273.1 Detailed Description . . . . .	3601
6.274 deque File Reference . . . . .	3601
6.274.1 Detailed Description . . . . .	3601
6.275 experimental/deque File Reference . . . . .	3601
6.275.1 Detailed Description . . . . .	3602
6.276 expected File Reference . . . . .	3602
6.276.1 Detailed Description . . . . .	3602
6.277 lfts_config.h File Reference . . . . .	3602
6.277.1 Detailed Description . . . . .	3602
6.278 ext/numeric_traits.h File Reference . . . . .	3602
6.278.1 Detailed Description . . . . .	3603
6.279 propagate_const File Reference . . . . .	3603
6.279.1 Detailed Description . . . . .	3604

6.280 simd File Reference . . . . .	3604
6.280.1 Detailed Description . . . . .	3604
6.281 aligned_buffer.h File Reference . . . . .	3604
6.281.1 Detailed Description . . . . .	3604
6.282 atomicity.h File Reference . . . . .	3605
6.282.1 Detailed Description . . . . .	3605
6.283 bitmap_allocator.h File Reference . . . . .	3605
6.283.1 Detailed Description . . . . .	3606
6.283.2 Macro Definition Documentation . . . . .	3606
6.284 cast.h File Reference . . . . .	3606
6.284.1 Detailed Description . . . . .	3607
6.285 codecvt_specializations.h File Reference . . . . .	3607
6.285.1 Detailed Description . . . . .	3607
6.286 concurrence.h File Reference . . . . .	3607
6.286.1 Detailed Description . . . . .	3608
6.287 debug_allocator.h File Reference . . . . .	3608
6.287.1 Detailed Description . . . . .	3608
6.288 enc_filebuf.h File Reference . . . . .	3608
6.288.1 Detailed Description . . . . .	3608
6.289 extptr_allocator.h File Reference . . . . .	3608
6.289.1 Detailed Description . . . . .	3609
6.290 malloc_allocator.h File Reference . . . . .	3609
6.290.1 Detailed Description . . . . .	3609
6.291 mt_allocator.h File Reference . . . . .	3609
6.291.1 Detailed Description . . . . .	3609
6.292 assoc_container.hpp File Reference . . . . .	3610
6.292.1 Detailed Description . . . . .	3610
6.293 bin_search_tree_.hpp File Reference . . . . .	3610
6.293.1 Detailed Description . . . . .	3610
6.294 bin_search_tree_/node_iterators.hpp File Reference . . . . .	3611
6.294.1 Detailed Description . . . . .	3611
6.295 ov_tree_map_/node_iterators.hpp File Reference . . . . .	3611
6.295.1 Detailed Description . . . . .	3611
6.296 point_iterators.hpp File Reference . . . . .	3611
6.296.1 Detailed Description . . . . .	3612
6.297 bin_search_tree_/r_erase_fn_imps.hpp File Reference . . . . .	3612
6.297.1 Detailed Description . . . . .	3612
6.298 pat_trie_/r_erase_fn_imps.hpp File Reference . . . . .	3612
6.298.1 Detailed Description . . . . .	3612

6.299 bin_search_tree_/rotate_fn_imps.hpp File Reference . . . . .	3612
6.299.1 Detailed Description . . . . .	3612
6.300 pat_trie_/rotate_fn_imps.hpp File Reference . . . . .	3612
6.300.1 Detailed Description . . . . .	3612
6.301 bin_search_tree_/traits.hpp File Reference . . . . .	3612
6.301.1 Detailed Description . . . . .	3612
6.302 branch_policy/traits.hpp File Reference . . . . .	3612
6.302.1 Detailed Description . . . . .	3612
6.303 ov_tree_map_/traits.hpp File Reference . . . . .	3613
6.303.1 Detailed Description . . . . .	3613
6.304 pat_trie_/traits.hpp File Reference . . . . .	3613
6.304.1 Detailed Description . . . . .	3613
6.305 rb_tree_map_/traits.hpp File Reference . . . . .	3613
6.305.1 Detailed Description . . . . .	3613
6.306 splay_tree_/traits.hpp File Reference . . . . .	3613
6.306.1 Detailed Description . . . . .	3614
6.307 binary_heap_.hpp File Reference . . . . .	3614
6.307.1 Detailed Description . . . . .	3614
6.308 binary_heap_/const_iterator.hpp File Reference . . . . .	3614
6.308.1 Detailed Description . . . . .	3614
6.309 left_child_next_sibling_heap_/const_iterator.hpp File Reference . . . . .	3615
6.309.1 Detailed Description . . . . .	3615
6.310 unordered_iterator/const_iterator.hpp File Reference . . . . .	3615
6.310.1 Detailed Description . . . . .	3615
6.311 bin_search_tree_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3615
6.311.1 Detailed Description . . . . .	3615
6.312 binary_heap_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3615
6.312.1 Detailed Description . . . . .	3615
6.313 binomial_heap_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3615
6.313.1 Detailed Description . . . . .	3615
6.314 binomial_heap_base_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3615
6.314.1 Detailed Description . . . . .	3615
6.315 left_child_next_sibling_heap_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3615
6.315.1 Detailed Description . . . . .	3615
6.316 ov_tree_map_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3616
6.316.1 Detailed Description . . . . .	3616
6.317 pairing_heap_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3616
6.317.1 Detailed Description . . . . .	3616
6.318 pat_trie_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3616

6.318.1 Detailed Description . . . . .	3616
6.319 rb_tree_map_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3616
6.319.1 Detailed Description . . . . .	3616
6.320 rc_binomial_heap_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3616
6.320.1 Detailed Description . . . . .	3616
6.321 splay_tree_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3616
6.321.1 Detailed Description . . . . .	3616
6.322 thin_heap_/constructors_destructor_fn_imps.hpp File Reference . . . . .	3616
6.322.1 Detailed Description . . . . .	3616
6.323 bin_search_tree_/debug_fn_imps.hpp File Reference . . . . .	3616
6.323.1 Detailed Description . . . . .	3616
6.324 binary_heap_/debug_fn_imps.hpp File Reference . . . . .	3616
6.324.1 Detailed Description . . . . .	3616
6.325 binomial_heap_/debug_fn_imps.hpp File Reference . . . . .	3616
6.325.1 Detailed Description . . . . .	3616
6.326 binomial_heap_base_/debug_fn_imps.hpp File Reference . . . . .	3617
6.326.1 Detailed Description . . . . .	3617
6.327 cc_hash_table_map_/debug_fn_imps.hpp File Reference . . . . .	3617
6.327.1 Detailed Description . . . . .	3617
6.328 gp_hash_table_map_/debug_fn_imps.hpp File Reference . . . . .	3617
6.328.1 Detailed Description . . . . .	3617
6.329 left_child_next_sibling_heap_/debug_fn_imps.hpp File Reference . . . . .	3617
6.329.1 Detailed Description . . . . .	3617
6.330 list_update_map_/debug_fn_imps.hpp File Reference . . . . .	3617
6.330.1 Detailed Description . . . . .	3617
6.331 ov_tree_map_/debug_fn_imps.hpp File Reference . . . . .	3617
6.331.1 Detailed Description . . . . .	3617
6.332 pairing_heap_/debug_fn_imps.hpp File Reference . . . . .	3617
6.332.1 Detailed Description . . . . .	3617
6.333 pat_trie_/debug_fn_imps.hpp File Reference . . . . .	3617
6.333.1 Detailed Description . . . . .	3617
6.334 rb_tree_map_/debug_fn_imps.hpp File Reference . . . . .	3617
6.334.1 Detailed Description . . . . .	3617
6.335 rc_binomial_heap_/debug_fn_imps.hpp File Reference . . . . .	3617
6.335.1 Detailed Description . . . . .	3617
6.336 splay_tree_/debug_fn_imps.hpp File Reference . . . . .	3618
6.336.1 Detailed Description . . . . .	3618
6.337 thin_heap_/debug_fn_imps.hpp File Reference . . . . .	3618
6.337.1 Detailed Description . . . . .	3618

6.338 entry_cmp.hpp File Reference	3618
6.338.1 Detailed Description	3618
6.339 entry_pred.hpp File Reference	3618
6.339.1 Detailed Description	3618
6.340 bin_search_tree_/erase_fn_imps.hpp File Reference	3618
6.340.1 Detailed Description	3618
6.341 binary_heap_/erase_fn_imps.hpp File Reference	3618
6.341.1 Detailed Description	3618
6.342 binomial_heap_base_/erase_fn_imps.hpp File Reference	3619
6.342.1 Detailed Description	3619
6.343 cc_hash_table_map_/erase_fn_imps.hpp File Reference	3619
6.343.1 Detailed Description	3619
6.344 gp_hash_table_map_/erase_fn_imps.hpp File Reference	3619
6.344.1 Detailed Description	3619
6.345 left_child_next_sibling_heap_/erase_fn_imps.hpp File Reference	3619
6.345.1 Detailed Description	3619
6.346 list_update_map_/erase_fn_imps.hpp File Reference	3619
6.346.1 Detailed Description	3619
6.347 ov_tree_map_/erase_fn_imps.hpp File Reference	3619
6.347.1 Detailed Description	3619
6.348 pairing_heap_/erase_fn_imps.hpp File Reference	3619
6.348.1 Detailed Description	3619
6.349 pat_trie_/erase_fn_imps.hpp File Reference	3619
6.349.1 Detailed Description	3619
6.350 rb_tree_map_/erase_fn_imps.hpp File Reference	3619
6.350.1 Detailed Description	3619
6.351 rc_binomial_heap_/erase_fn_imps.hpp File Reference	3619
6.351.1 Detailed Description	3619
6.352 splay_tree_/erase_fn_imps.hpp File Reference	3620
6.352.1 Detailed Description	3620
6.353 thin_heap_/erase_fn_imps.hpp File Reference	3620
6.353.1 Detailed Description	3620
6.354 bin_search_tree_/find_fn_imps.hpp File Reference	3620
6.354.1 Detailed Description	3620
6.355 binary_heap_/find_fn_imps.hpp File Reference	3620
6.355.1 Detailed Description	3620
6.356 binomial_heap_base_/find_fn_imps.hpp File Reference	3620
6.356.1 Detailed Description	3620
6.357 cc_hash_table_map_/find_fn_imps.hpp File Reference	3620



6.357.1 Detailed Description	3620
6.358 gp_hash_table_map_/find_fn_imps.hpp File Reference	3620
6.358.1 Detailed Description	3620
6.359 list_update_map_/find_fn_imps.hpp File Reference	3620
6.359.1 Detailed Description	3620
6.360 pairing_heap_/find_fn_imps.hpp File Reference	3620
6.360.1 Detailed Description	3620
6.361 pat_trie_/find_fn_imps.hpp File Reference	3620
6.361.1 Detailed Description	3620
6.362 rb_tree_map_/find_fn_imps.hpp File Reference	3621
6.362.1 Detailed Description	3621
6.363 splay_tree_/find_fn_imps.hpp File Reference	3621
6.363.1 Detailed Description	3621
6.364 thin_heap_/find_fn_imps.hpp File Reference	3621
6.364.1 Detailed Description	3621
6.365 bin_search_tree_/info_fn_imps.hpp File Reference	3621
6.365.1 Detailed Description	3621
6.366 binary_heap_/info_fn_imps.hpp File Reference	3621
6.366.1 Detailed Description	3621
6.367 cc_hash_table_map_/info_fn_imps.hpp File Reference	3621
6.367.1 Detailed Description	3621
6.368 gp_hash_table_map_/info_fn_imps.hpp File Reference	3621
6.368.1 Detailed Description	3621
6.369 left_child_next_sibling_heap_/info_fn_imps.hpp File Reference	3621
6.369.1 Detailed Description	3621
6.370 list_update_map_/info_fn_imps.hpp File Reference	3621
6.370.1 Detailed Description	3621
6.371 ov_tree_map_/info_fn_imps.hpp File Reference	3621
6.371.1 Detailed Description	3621
6.372 pat_trie_/info_fn_imps.hpp File Reference	3622
6.372.1 Detailed Description	3622
6.373 rb_tree_map_/info_fn_imps.hpp File Reference	3622
6.373.1 Detailed Description	3622
6.374 splay_tree_/info_fn_imps.hpp File Reference	3622
6.374.1 Detailed Description	3622
6.375 bin_search_tree_/insert_fn_imps.hpp File Reference	3622
6.375.1 Detailed Description	3622
6.376 binary_heap_/insert_fn_imps.hpp File Reference	3622
6.376.1 Detailed Description	3622

6.377 binomial_heap_base_/insert_fn_imps.hpp File Reference . . . . .	3622
6.377.1 Detailed Description . . . . .	3622
6.378 cc_hash_table_map_/insert_fn_imps.hpp File Reference . . . . .	3622
6.378.1 Detailed Description . . . . .	3622
6.379 gp_hash_table_map_/insert_fn_imps.hpp File Reference . . . . .	3622
6.379.1 Detailed Description . . . . .	3622
6.380 left_child_next_sibling_heap_/insert_fn_imps.hpp File Reference . . . . .	3622
6.380.1 Detailed Description . . . . .	3622
6.381 list_update_map_/insert_fn_imps.hpp File Reference . . . . .	3622
6.381.1 Detailed Description . . . . .	3622
6.382 ov_tree_map_/insert_fn_imps.hpp File Reference . . . . .	3623
6.382.1 Detailed Description . . . . .	3623
6.383 pairing_heap_/insert_fn_imps.hpp File Reference . . . . .	3623
6.383.1 Detailed Description . . . . .	3623
6.384 rb_tree_map_/insert_fn_imps.hpp File Reference . . . . .	3623
6.384.1 Detailed Description . . . . .	3623
6.385 rc_binomial_heap_/insert_fn_imps.hpp File Reference . . . . .	3623
6.385.1 Detailed Description . . . . .	3623
6.386 splay_tree_/insert_fn_imps.hpp File Reference . . . . .	3623
6.386.1 Detailed Description . . . . .	3623
6.387 thin_heap_/insert_fn_imps.hpp File Reference . . . . .	3623
6.387.1 Detailed Description . . . . .	3623
6.388 bin_search_tree_/iterators_fn_imps.hpp File Reference . . . . .	3623
6.388.1 Detailed Description . . . . .	3623
6.389 binary_heap_/iterators_fn_imps.hpp File Reference . . . . .	3623
6.389.1 Detailed Description . . . . .	3623
6.390 cc_hash_table_map_/iterators_fn_imps.hpp File Reference . . . . .	3623
6.390.1 Detailed Description . . . . .	3623
6.391 left_child_next_sibling_heap_/iterators_fn_imps.hpp File Reference . . . . .	3623
6.391.1 Detailed Description . . . . .	3623
6.392 list_update_map_/iterators_fn_imps.hpp File Reference . . . . .	3624
6.392.1 Detailed Description . . . . .	3624
6.393 ov_tree_map_/iterators_fn_imps.hpp File Reference . . . . .	3624
6.393.1 Detailed Description . . . . .	3624
6.394 pat_trie_/iterators_fn_imps.hpp File Reference . . . . .	3624
6.394.1 Detailed Description . . . . .	3624
6.395 binary_heap_/point_const_iterator.hpp File Reference . . . . .	3624
6.395.1 Detailed Description . . . . .	3624
6.396 left_child_next_sibling_heap_/point_const_iterator.hpp File Reference . . . . .	3624

6.396.1 Detailed Description . . . . .	3624
6.397 unordered_iterator/point_const_iterator.hpp File Reference . . . . .	3624
6.397.1 Detailed Description . . . . .	3624
6.398 bin_search_tree_/policy_access_fn_imps.hpp File Reference . . . . .	3625
6.398.1 Detailed Description . . . . .	3625
6.399 binary_heap_/policy_access_fn_imps.hpp File Reference . . . . .	3625
6.399.1 Detailed Description . . . . .	3625
6.400 cc_hash_table_map_/policy_access_fn_imps.hpp File Reference . . . . .	3625
6.400.1 Detailed Description . . . . .	3625
6.401 gp_hash_table_map_/policy_access_fn_imps.hpp File Reference . . . . .	3625
6.401.1 Detailed Description . . . . .	3625
6.402 left_child_next_sibling_heap_/policy_access_fn_imps.hpp File Reference . . . . .	3625
6.402.1 Detailed Description . . . . .	3625
6.403 ov_tree_map_/policy_access_fn_imps.hpp File Reference . . . . .	3625
6.403.1 Detailed Description . . . . .	3625
6.404 pat_trie_/policy_access_fn_imps.hpp File Reference . . . . .	3625
6.404.1 Detailed Description . . . . .	3625
6.405 resize_policy.hpp File Reference . . . . .	3625
6.405.1 Detailed Description . . . . .	3626
6.406 bin_search_tree_/split_join_fn_imps.hpp File Reference . . . . .	3626
6.406.1 Detailed Description . . . . .	3626
6.407 binary_heap_/split_join_fn_imps.hpp File Reference . . . . .	3626
6.407.1 Detailed Description . . . . .	3626
6.408 binomial_heap_base_/split_join_fn_imps.hpp File Reference . . . . .	3626
6.408.1 Detailed Description . . . . .	3626
6.409 ov_tree_map_/split_join_fn_imps.hpp File Reference . . . . .	3626
6.409.1 Detailed Description . . . . .	3626
6.410 pairing_heap_/split_join_fn_imps.hpp File Reference . . . . .	3626
6.410.1 Detailed Description . . . . .	3626
6.411 rb_tree_map_/split_join_fn_imps.hpp File Reference . . . . .	3626
6.411.1 Detailed Description . . . . .	3626
6.412 rc_binomial_heap_/split_join_fn_imps.hpp File Reference . . . . .	3626
6.412.1 Detailed Description . . . . .	3626
6.413 splay_tree_/split_join_fn_imps.hpp File Reference . . . . .	3626
6.413.1 Detailed Description . . . . .	3626
6.414 thin_heap_/split_join_fn_imps.hpp File Reference . . . . .	3626
6.414.1 Detailed Description . . . . .	3626
6.415 binary_heap_/trace_fn_imps.hpp File Reference . . . . .	3627
6.415.1 Detailed Description . . . . .	3627

6.416 cc_hash_table_map_/trace_fn_imps.hpp File Reference . . . . .	3627
6.416.1 Detailed Description . . . . .	3627
6.417 gp_hash_table_map_/trace_fn_imps.hpp File Reference . . . . .	3627
6.417.1 Detailed Description . . . . .	3627
6.418 left_child_next_sibling_heap_/trace_fn_imps.hpp File Reference . . . . .	3627
6.418.1 Detailed Description . . . . .	3627
6.419 list_update_map_/trace_fn_imps.hpp File Reference . . . . .	3627
6.419.1 Detailed Description . . . . .	3627
6.420 pat_trie_/trace_fn_imps.hpp File Reference . . . . .	3627
6.420.1 Detailed Description . . . . .	3627
6.421 rc_binomial_heap_/trace_fn_imps.hpp File Reference . . . . .	3627
6.421.1 Detailed Description . . . . .	3627
6.422 thin_heap_/trace_fn_imps.hpp File Reference . . . . .	3627
6.422.1 Detailed Description . . . . .	3627
6.423 binomial_heap_.hpp File Reference . . . . .	3627
6.423.1 Detailed Description . . . . .	3628
6.424 binomial_heap_base_.hpp File Reference . . . . .	3628
6.424.1 Detailed Description . . . . .	3628
6.425 branch_policy.hpp File Reference . . . . .	3628
6.425.1 Detailed Description . . . . .	3628
6.426 null_node_metadata.hpp File Reference . . . . .	3628
6.426.1 Detailed Description . . . . .	3629
6.427 cc_ht_map_.hpp File Reference . . . . .	3629
6.427.1 Detailed Description . . . . .	3629
6.428 cmp_fn_imps.hpp File Reference . . . . .	3629
6.428.1 Detailed Description . . . . .	3629
6.429 cond_key_dtor_entry_dealtor.hpp File Reference . . . . .	3630
6.429.1 Detailed Description . . . . .	3630
6.430 cc_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference . . . . .	3630
6.430.1 Detailed Description . . . . .	3630
6.431 gp_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference . . . . .	3630
6.431.1 Detailed Description . . . . .	3630
6.432 list_update_map_/constructor_destructor_fn_imps.hpp File Reference . . . . .	3630
6.433 cc_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference . . . . .	3630
6.433.1 Detailed Description . . . . .	3630
6.434 gp_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference . . . . .	3630
6.434.1 Detailed Description . . . . .	3630
6.435 cc_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference . . . . .	3630
6.435.1 Detailed Description . . . . .	3630

6.436	<a href="#">gp_hash_table_map /constructor_destructor_store_hash_fn_imps.hpp File Reference</a>	3630
6.436.1	Detailed Description	3630
6.437	<a href="#">cc_hash_table_map /debug_no_store_hash_fn_imps.hpp File Reference</a>	3631
6.437.1	Detailed Description	3631
6.438	<a href="#">gp_hash_table_map /debug_no_store_hash_fn_imps.hpp File Reference</a>	3631
6.438.1	Detailed Description	3631
6.439	<a href="#">cc_hash_table_map /debug_store_hash_fn_imps.hpp File Reference</a>	3631
6.439.1	Detailed Description	3631
6.440	<a href="#">gp_hash_table_map /debug_store_hash_fn_imps.hpp File Reference</a>	3631
6.440.1	Detailed Description	3631
6.441	<a href="#">entry_list_fn_imps.hpp File Reference</a>	3631
6.441.1	Detailed Description	3631
6.442	<a href="#">cc_hash_table_map /erase_no_store_hash_fn_imps.hpp File Reference</a>	3631
6.442.1	Detailed Description	3631
6.443	<a href="#">gp_hash_table_map /erase_no_store_hash_fn_imps.hpp File Reference</a>	3631
6.443.1	Detailed Description	3631
6.444	<a href="#">cc_hash_table_map /erase_store_hash_fn_imps.hpp File Reference</a>	3631
6.444.1	Detailed Description	3631
6.445	<a href="#">gp_hash_table_map /erase_store_hash_fn_imps.hpp File Reference</a>	3631
6.445.1	Detailed Description	3631
6.446	<a href="#">cc_hash_table_map /find_store_hash_fn_imps.hpp File Reference</a>	3631
6.446.1	Detailed Description	3631
6.447	<a href="#">gp_hash_table_map /find_store_hash_fn_imps.hpp File Reference</a>	3632
6.447.1	Detailed Description	3632
6.448	<a href="#">cc_hash_table_map /insert_no_store_hash_fn_imps.hpp File Reference</a>	3632
6.448.1	Detailed Description	3632
6.449	<a href="#">gp_hash_table_map /insert_no_store_hash_fn_imps.hpp File Reference</a>	3632
6.449.1	Detailed Description	3632
6.450	<a href="#">cc_hash_table_map /insert_store_hash_fn_imps.hpp File Reference</a>	3632
6.450.1	Detailed Description	3632
6.451	<a href="#">gp_hash_table_map /insert_store_hash_fn_imps.hpp File Reference</a>	3632
6.451.1	Detailed Description	3632
6.452	<a href="#">cc_hash_table_map /resize_fn_imps.hpp File Reference</a>	3632
6.452.1	Detailed Description	3632
6.453	<a href="#">gp_hash_table_map /resize_fn_imps.hpp File Reference</a>	3632
6.453.1	Detailed Description	3632
6.454	<a href="#">cc_hash_table_map /resize_no_store_hash_fn_imps.hpp File Reference</a>	3632
6.454.1	Detailed Description	3632
6.455	<a href="#">gp_hash_table_map /resize_no_store_hash_fn_imps.hpp File Reference</a>	3632

6.455.1 Detailed Description . . . . .	3632
6.456 cc_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference . . . . .	3632
6.456.1 Detailed Description . . . . .	3632
6.457 gp_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference . . . . .	3633
6.457.1 Detailed Description . . . . .	3633
6.458 size_fn_imps.hpp File Reference . . . . .	3633
6.458.1 Detailed Description . . . . .	3633
6.459 cond_dealtor.hpp File Reference . . . . .	3633
6.459.1 Detailed Description . . . . .	3633
6.460 container_base_dispatch.hpp File Reference . . . . .	3633
6.460.1 Detailed Description . . . . .	3634
6.461 debug_map_base.hpp File Reference . . . . .	3634
6.461.1 Detailed Description . . . . .	3634
6.462 eq_by_less.hpp File Reference . . . . .	3634
6.462.1 Detailed Description . . . . .	3634
6.463 hash_eq_fn.hpp File Reference . . . . .	3634
6.463.1 Detailed Description . . . . .	3634
6.464 find_no_store_hash_fn_imps.hpp File Reference . . . . .	3635
6.464.1 Detailed Description . . . . .	3635
6.465 gp_ht_map_.hpp File Reference . . . . .	3635
6.465.1 Detailed Description . . . . .	3635
6.466 iterator_fn_imps.hpp File Reference . . . . .	3635
6.466.1 Detailed Description . . . . .	3635
6.467 direct_mask_range_hashing_imp.hpp File Reference . . . . .	3636
6.467.1 Detailed Description . . . . .	3636
6.468 direct_mod_range_hashing_imp.hpp File Reference . . . . .	3636
6.468.1 Detailed Description . . . . .	3636
6.469 linear_probe_fn_imp.hpp File Reference . . . . .	3636
6.469.1 Detailed Description . . . . .	3636
6.470 mask_based_range_hashing.hpp File Reference . . . . .	3636
6.470.1 Detailed Description . . . . .	3636
6.471 mod_based_range_hashing.hpp File Reference . . . . .	3636
6.471.1 Detailed Description . . . . .	3636
6.472 probe_fn_base.hpp File Reference . . . . .	3637
6.472.1 Detailed Description . . . . .	3637
6.473 quadratic_probe_fn_imp.hpp File Reference . . . . .	3637
6.473.1 Detailed Description . . . . .	3637
6.474 ranged_hash_fn.hpp File Reference . . . . .	3637
6.474.1 Detailed Description . . . . .	3637

6.475 ranged_probe_fn.hpp File Reference . . . . .	3637
6.475.1 Detailed Description . . . . .	3638
6.476 sample_probe_fn.hpp File Reference . . . . .	3638
6.476.1 Detailed Description . . . . .	3638
6.477 sample_range_hashing.hpp File Reference . . . . .	3638
6.477.1 Detailed Description . . . . .	3638
6.478 sample_ranged_hash_fn.hpp File Reference . . . . .	3638
6.478.1 Detailed Description . . . . .	3638
6.479 sample_ranged_probe_fn.hpp File Reference . . . . .	3639
6.479.1 Detailed Description . . . . .	3639
6.480 left_child_next_sibling_heap_.hpp File Reference . . . . .	3639
6.480.1 Detailed Description . . . . .	3639
6.481 left_child_next_sibling_heap_/node.hpp File Reference . . . . .	3639
6.481.1 Detailed Description . . . . .	3639
6.482 rb_tree_map_/node.hpp File Reference . . . . .	3640
6.482.1 Detailed Description . . . . .	3640
6.483 splay_tree_/node.hpp File Reference . . . . .	3640
6.483.1 Detailed Description . . . . .	3640
6.484 entry_metadata_base.hpp File Reference . . . . .	3640
6.484.1 Detailed Description . . . . .	3640
6.485 lu_map_.hpp File Reference . . . . .	3640
6.485.1 Detailed Description . . . . .	3641
6.486 lu_counter_metadata.hpp File Reference . . . . .	3641
6.486.1 Detailed Description . . . . .	3641
6.487 sample_update_policy.hpp File Reference . . . . .	3641
6.487.1 Detailed Description . . . . .	3641
6.488 ov_tree_map_.hpp File Reference . . . . .	3641
6.488.1 Detailed Description . . . . .	3642
6.489 pairing_heap_.hpp File Reference . . . . .	3642
6.489.1 Detailed Description . . . . .	3642
6.490 insert_join_fnimps.hpp File Reference . . . . .	3642
6.490.1 Detailed Description . . . . .	3642
6.491 pat_trie_.hpp File Reference . . . . .	3642
6.491.1 Detailed Description . . . . .	3643
6.492 pat_trie_base.hpp File Reference . . . . .	3643
6.492.1 Detailed Description . . . . .	3644
6.493 split_fnimps.hpp File Reference . . . . .	3644
6.493.1 Detailed Description . . . . .	3644
6.494 synth_access_traits.hpp File Reference . . . . .	3644

6.494.1 Detailed Description . . . . .	3644
6.495 update_fn_imps.hpp File Reference . . . . .	3644
6.495.1 Detailed Description . . . . .	3644
6.496 priority_queue_base_dispatch.hpp File Reference . . . . .	3645
6.496.1 Detailed Description . . . . .	3645
6.497 rb_tree_.hpp File Reference . . . . .	3645
6.497.1 Detailed Description . . . . .	3645
6.498 rc.hpp File Reference . . . . .	3645
6.498.1 Detailed Description . . . . .	3646
6.499 rc_binomial_heap_.hpp File Reference . . . . .	3646
6.499.1 Detailed Description . . . . .	3646
6.500 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference . . . . .	3646
6.500.1 Detailed Description . . . . .	3646
6.501 hash_exponential_size_policy_imp.hpp File Reference . . . . .	3646
6.501.1 Detailed Description . . . . .	3646
6.502 hash_load_check_resize_trigger_imp.hpp File Reference . . . . .	3646
6.502.1 Detailed Description . . . . .	3646
6.503 hash_load_check_resize_trigger_size_base.hpp File Reference . . . . .	3646
6.503.1 Detailed Description . . . . .	3646
6.504 hash_prime_size_policy_imp.hpp File Reference . . . . .	3647
6.504.1 Detailed Description . . . . .	3647
6.505 hash_standard_resize_policy_imp.hpp File Reference . . . . .	3647
6.505.1 Detailed Description . . . . .	3647
6.506 sample_resize_policy.hpp File Reference . . . . .	3647
6.506.1 Detailed Description . . . . .	3647
6.507 sample_resize_trigger.hpp File Reference . . . . .	3647
6.507.1 Detailed Description . . . . .	3647
6.508 sample_size_policy.hpp File Reference . . . . .	3647
6.508.1 Detailed Description . . . . .	3647
6.509 splay_fn_imps.hpp File Reference . . . . .	3648
6.509.1 Detailed Description . . . . .	3648
6.510 splay_tree_.hpp File Reference . . . . .	3648
6.510.1 Detailed Description . . . . .	3648
6.511 standard_policies.hpp File Reference . . . . .	3648
6.511.1 Detailed Description . . . . .	3649
6.511.2 Enumeration Type Documentation . . . . .	3649
6.512 thin_heap_.hpp File Reference . . . . .	3649
6.512.1 Detailed Description . . . . .	3649
6.513 tree_policy/node_metadata_selector.hpp File Reference . . . . .	3649



6.513.1 Detailed Description . . . . .	3649
6.514 trie_policy/node_metadata_selector.hpp File Reference . . . . .	3650
6.514.1 Detailed Description . . . . .	3650
6.515 tree_policy/order_statistics_imp.hpp File Reference . . . . .	3650
6.515.1 Detailed Description . . . . .	3650
6.516 trie_policy/order_statistics_imp.hpp File Reference . . . . .	3650
6.516.1 Detailed Description . . . . .	3650
6.517 sample_tree_node_update.hpp File Reference . . . . .	3650
6.517.1 Detailed Description . . . . .	3650
6.518 tree_trace_base.hpp File Reference . . . . .	3650
6.518.1 Detailed Description . . . . .	3650
6.519 prefix_search_node_update_imp.hpp File Reference . . . . .	3650
6.519.1 Detailed Description . . . . .	3650
6.520 sample_trie_access_traits.hpp File Reference . . . . .	3651
6.520.1 Detailed Description . . . . .	3651
6.521 sample_trie_node_update.hpp File Reference . . . . .	3651
6.521.1 Detailed Description . . . . .	3651
6.522 trie_policy_base.hpp File Reference . . . . .	3651
6.522.1 Detailed Description . . . . .	3651
6.523 trie_string_access_traits_imp.hpp File Reference . . . . .	3651
6.523.1 Detailed Description . . . . .	3651
6.524 type_utils.hpp File Reference . . . . .	3651
6.524.1 Detailed Description . . . . .	3652
6.525 types_traits.hpp File Reference . . . . .	3652
6.525.1 Detailed Description . . . . .	3652
6.526 iterator.hpp File Reference . . . . .	3652
6.526.1 Detailed Description . . . . .	3652
6.527 point_iterator.hpp File Reference . . . . .	3653
6.527.1 Detailed Description . . . . .	3653
6.528 exception.hpp File Reference . . . . .	3653
6.528.1 Detailed Description . . . . .	3653
6.529 hash_policy.hpp File Reference . . . . .	3653
6.529.1 Detailed Description . . . . .	3654
6.530 list_update_policy.hpp File Reference . . . . .	3654
6.530.1 Detailed Description . . . . .	3654
6.531 priority_queue.hpp File Reference . . . . .	3654
6.531.1 Detailed Description . . . . .	3655
6.532 tag_and_trait.hpp File Reference . . . . .	3655
6.532.1 Detailed Description . . . . .	3656

6.533 tree_policy.hpp File Reference . . . . .	3656
6.533.1 Detailed Description . . . . .	3656
6.534 trie_policy.hpp File Reference . . . . .	3656
6.534.1 Detailed Description . . . . .	3657
6.535 pod_char_traits.h File Reference . . . . .	3657
6.535.1 Detailed Description . . . . .	3657
6.536 pointer.h File Reference . . . . .	3657
6.536.1 Detailed Description . . . . .	3659
6.537 pool_allocator.h File Reference . . . . .	3659
6.537.1 Detailed Description . . . . .	3659
6.538 rb_tree File Reference . . . . .	3659
6.538.1 Detailed Description . . . . .	3660
6.539 rc_string_base.h File Reference . . . . .	3660
6.539.1 Detailed Description . . . . .	3660
6.540 rope File Reference . . . . .	3660
6.540.1 Detailed Description . . . . .	3663
6.541 ropeimpl.h File Reference . . . . .	3663
6.541.1 Detailed Description . . . . .	3670
6.542 slist File Reference . . . . .	3670
6.542.1 Detailed Description . . . . .	3671
6.543 sso_string_base.h File Reference . . . . .	3671
6.543.1 Detailed Description . . . . .	3671
6.544 stdio_filebuf.h File Reference . . . . .	3671
6.544.1 Detailed Description . . . . .	3671
6.545 stdio_sync_filebuf.h File Reference . . . . .	3671
6.545.1 Detailed Description . . . . .	3672
6.546 string_conversions.h File Reference . . . . .	3672
6.546.1 Detailed Description . . . . .	3672
6.547 throw_allocator.h File Reference . . . . .	3672
6.547.1 Detailed Description . . . . .	3673
6.548 type_traits.h File Reference . . . . .	3673
6.548.1 Detailed Description . . . . .	3673
6.549 typelist.h File Reference . . . . .	3673
6.549.1 Detailed Description . . . . .	3674
6.550 vstring.h File Reference . . . . .	3674
6.550.1 Detailed Description . . . . .	3677
6.551 vstring.tcc File Reference . . . . .	3677
6.551.1 Detailed Description . . . . .	3678
6.552 vstring_fwd.h File Reference . . . . .	3678

6.552.1 Detailed Description . . . . .	3679
6.553 vstring_util.h File Reference . . . . .	3679
6.553.1 Detailed Description . . . . .	3679
6.554 fenv.h File Reference . . . . .	3679
6.554.1 Detailed Description . . . . .	3679
6.555 experimental/filesystem File Reference . . . . .	3679
6.555.1 Detailed Description . . . . .	3679
6.556 filesystem File Reference . . . . .	3679
6.556.1 Detailed Description . . . . .	3679
6.557 flat_map File Reference . . . . .	3680
6.557.1 Detailed Description . . . . .	3680
6.558 flat_set File Reference . . . . .	3680
6.558.1 Detailed Description . . . . .	3680
6.559 format File Reference . . . . .	3680
6.559.1 Detailed Description . . . . .	3680
6.560 debug/forward_list File Reference . . . . .	3680
6.560.1 Detailed Description . . . . .	3681
6.561 experimental/forward_list File Reference . . . . .	3681
6.561.1 Detailed Description . . . . .	3681
6.562 forward_list File Reference . . . . .	3681
6.562.1 Detailed Description . . . . .	3682
6.563 fstream File Reference . . . . .	3682
6.563.1 Detailed Description . . . . .	3683
6.564 experimental/functional File Reference . . . . .	3683
6.564.1 Detailed Description . . . . .	3683
6.565 ext/functional File Reference . . . . .	3684
6.565.1 Detailed Description . . . . .	3684
6.566 functional File Reference . . . . .	3685
6.566.1 Detailed Description . . . . .	3686
6.567 future File Reference . . . . .	3687
6.567.1 Detailed Description . . . . .	3688
6.568 generator File Reference . . . . .	3688
6.568.1 Detailed Description . . . . .	3688
6.569 inplace_vector File Reference . . . . .	3688
6.569.1 Detailed Description . . . . .	3688
6.570 iomanip File Reference . . . . .	3688
6.570.1 Detailed Description . . . . .	3690
6.571 ios File Reference . . . . .	3690
6.571.1 Detailed Description . . . . .	3690

6.572 iosfwd File Reference . . . . .	3690
6.572.1 Detailed Description . . . . .	3691
6.573 iostream File Reference . . . . .	3691
6.573.1 Detailed Description . . . . .	3691
6.574 istream File Reference . . . . .	3692
6.574.1 Detailed Description . . . . .	3692
6.575 experimental/iterator File Reference . . . . .	3693
6.575.1 Detailed Description . . . . .	3693
6.576 ext/iterator File Reference . . . . .	3693
6.576.1 Detailed Description . . . . .	3693
6.577 iterator File Reference . . . . .	3693
6.577.1 Detailed Description . . . . .	3694
6.578 latch File Reference . . . . .	3694
6.578.1 Detailed Description . . . . .	3694
6.579 limits File Reference . . . . .	3694
6.579.1 Detailed Description . . . . .	3695
6.580 debug/list File Reference . . . . .	3695
6.580.1 Detailed Description . . . . .	3696
6.581 experimental/list File Reference . . . . .	3696
6.581.1 Detailed Description . . . . .	3696
6.582 list File Reference . . . . .	3696
6.582.1 Detailed Description . . . . .	3697
6.583 locale File Reference . . . . .	3697
6.583.1 Detailed Description . . . . .	3697
6.584 debug/map File Reference . . . . .	3697
6.584.1 Detailed Description . . . . .	3697
6.585 experimental/map File Reference . . . . .	3697
6.585.1 Detailed Description . . . . .	3698
6.586 map File Reference . . . . .	3698
6.586.1 Detailed Description . . . . .	3698
6.587 math.h File Reference . . . . .	3699
6.587.1 Detailed Description . . . . .	3700
6.587.2 Function Documentation . . . . .	3700
6.588 mdspan File Reference . . . . .	3702
6.588.1 Detailed Description . . . . .	3702
6.589 experimental/memory File Reference . . . . .	3702
6.589.1 Detailed Description . . . . .	3703
6.590 ext/memory File Reference . . . . .	3703
6.590.1 Detailed Description . . . . .	3704

6.591 memory File Reference . . . . .	3704
6.591.1 Detailed Description . . . . .	3705
6.592 experimental/memory_resource File Reference . . . . .	3705
6.592.1 Detailed Description . . . . .	3705
6.592.2 Function Documentation . . . . .	3706
6.593 memory_resource File Reference . . . . .	3706
6.593.1 Detailed Description . . . . .	3706
6.593.2 Function Documentation . . . . .	3706
6.594 mutex File Reference . . . . .	3707
6.594.1 Detailed Description . . . . .	3707
6.595 numbers File Reference . . . . .	3707
6.595.1 Detailed Description . . . . .	3707
6.596 experimental/numeric File Reference . . . . .	3707
6.596.1 Detailed Description . . . . .	3708
6.597 ext/numeric File Reference . . . . .	3708
6.597.1 Detailed Description . . . . .	3708
6.598 numeric File Reference . . . . .	3708
6.598.1 Detailed Description . . . . .	3710
6.599 parallel/numeric File Reference . . . . .	3710
6.599.1 Detailed Description . . . . .	3712
6.600 experimental/optional File Reference . . . . .	3712
6.600.1 Detailed Description . . . . .	3712
6.601 optional File Reference . . . . .	3712
6.601.1 Detailed Description . . . . .	3712
6.602 ostream File Reference . . . . .	3712
6.602.1 Detailed Description . . . . .	3713
6.603 algo.h File Reference . . . . .	3713
6.603.1 Detailed Description . . . . .	3722
6.604 algbase.h File Reference . . . . .	3722
6.604.1 Detailed Description . . . . .	3724
6.605 balanced_quicksort.h File Reference . . . . .	3724
6.605.1 Detailed Description . . . . .	3725
6.606 base.h File Reference . . . . .	3725
6.606.1 Detailed Description . . . . .	3725
6.607 basic_iterator.h File Reference . . . . .	3726
6.607.1 Detailed Description . . . . .	3726
6.608 checkers.h File Reference . . . . .	3726
6.608.1 Detailed Description . . . . .	3726
6.609 compiletime_settings.h File Reference . . . . .	3726

---

6.609.1 Detailed Description	3726
6.609.2 Macro Definition Documentation	3726
6.610 equally_split.h File Reference	3727
6.610.1 Detailed Description	3728
6.611 features.h File Reference	3728
6.611.1 Detailed Description	3728
6.611.2 Macro Definition Documentation	3728
6.612 find.h File Reference	3730
6.612.1 Detailed Description	3730
6.613 find_selectors.h File Reference	3730
6.613.1 Detailed Description	3730
6.614 for_each.h File Reference	3730
6.614.1 Detailed Description	3731
6.615 for_each_selectors.h File Reference	3731
6.615.1 Detailed Description	3731
6.616 iterator.h File Reference	3731
6.616.1 Detailed Description	3731
6.617 list_partition.h File Reference	3732
6.617.1 Detailed Description	3732
6.618 losertree.h File Reference	3732
6.618.1 Detailed Description	3732
6.619 merge.h File Reference	3732
6.619.1 Detailed Description	3733
6.620 multiseq_selection.h File Reference	3733
6.620.1 Detailed Description	3733
6.621 multiway_merge.h File Reference	3734
6.621.1 Detailed Description	3736
6.621.2 Macro Definition Documentation	3736
6.622 multiway_mergesort.h File Reference	3736
6.622.1 Detailed Description	3737
6.623 numericfwd.h File Reference	3737
6.623.1 Detailed Description	3738
6.624 omp_loop.h File Reference	3738
6.624.1 Detailed Description	3739
6.625 omp_loop_static.h File Reference	3739
6.625.1 Detailed Description	3739
6.626 par_loop.h File Reference	3739
6.626.1 Detailed Description	3739
6.627 parallel.h File Reference	3739

6.627.1 Detailed Description . . . . .	3739
6.628 partial_sum.h File Reference . . . . .	3740
6.628.1 Detailed Description . . . . .	3740
6.629 partition.h File Reference . . . . .	3740
6.629.1 Detailed Description . . . . .	3740
6.629.2 Macro Definition Documentation . . . . .	3740
6.630 queue.h File Reference . . . . .	3741
6.630.1 Detailed Description . . . . .	3741
6.630.2 Macro Definition Documentation . . . . .	3741
6.631 quicksort.h File Reference . . . . .	3741
6.631.1 Detailed Description . . . . .	3741
6.632 random_number.h File Reference . . . . .	3741
6.632.1 Detailed Description . . . . .	3742
6.633 random_shuffle.h File Reference . . . . .	3742
6.633.1 Detailed Description . . . . .	3742
6.634 search.h File Reference . . . . .	3742
6.634.1 Detailed Description . . . . .	3743
6.635 set_operations.h File Reference . . . . .	3743
6.635.1 Detailed Description . . . . .	3743
6.636 settings.h File Reference . . . . .	3743
6.636.1 Detailed Description . . . . .	3744
6.636.2 Deciding whether to run an algorithm in parallel. . . . .	3744
6.636.3 Macro Definition Documentation . . . . .	3744
6.637 sort.h File Reference . . . . .	3744
6.637.1 Detailed Description . . . . .	3745
6.638 tags.h File Reference . . . . .	3745
6.638.1 Detailed Description . . . . .	3746
6.639 types.h File Reference . . . . .	3746
6.639.1 Detailed Description . . . . .	3746
6.640 unique_copy.h File Reference . . . . .	3746
6.640.1 Detailed Description . . . . .	3747
6.641 workstealing.h File Reference . . . . .	3747
6.641.1 Detailed Description . . . . .	3747
6.642 print File Reference . . . . .	3747
6.642.1 Detailed Description . . . . .	3747
6.643 queue File Reference . . . . .	3747
6.643.1 Detailed Description . . . . .	3747
6.644 experimental/random File Reference . . . . .	3748
6.644.1 Detailed Description . . . . .	3748

6.645 random File Reference . . . . .	3748
6.645.1 Detailed Description . . . . .	3748
6.646 ranges File Reference . . . . .	3748
6.646.1 Detailed Description . . . . .	3748
6.647 experimental/ratio File Reference . . . . .	3748
6.647.1 Detailed Description . . . . .	3749
6.648 ratio File Reference . . . . .	3749
6.648.1 Detailed Description . . . . .	3750
6.649 tr2/ratio File Reference . . . . .	3750
6.649.1 Detailed Description . . . . .	3750
6.650 experimental/regex File Reference . . . . .	3750
6.650.1 Detailed Description . . . . .	3751
6.651 regex File Reference . . . . .	3751
6.651.1 Detailed Description . . . . .	3751
6.652 scoped_allocator File Reference . . . . .	3751
6.652.1 Detailed Description . . . . .	3751
6.653 semaphore File Reference . . . . .	3752
6.653.1 Detailed Description . . . . .	3752
6.654 debug/set File Reference . . . . .	3752
6.654.1 Detailed Description . . . . .	3752
6.655 experimental/set File Reference . . . . .	3752
6.655.1 Detailed Description . . . . .	3752
6.656 set File Reference . . . . .	3753
6.656.1 Detailed Description . . . . .	3753
6.657 shared_mutex File Reference . . . . .	3753
6.657.1 Detailed Description . . . . .	3753
6.658 source_location File Reference . . . . .	3754
6.658.1 Detailed Description . . . . .	3754
6.659 span File Reference . . . . .	3754
6.659.1 Detailed Description . . . . .	3754
6.660 sstream File Reference . . . . .	3754
6.660.1 Detailed Description . . . . .	3755
6.661 stack File Reference . . . . .	3755
6.661.1 Detailed Description . . . . .	3755
6.662 stdatomic.h File Reference . . . . .	3755
6.662.1 Detailed Description . . . . .	3755
6.663 stdbit.h File Reference . . . . .	3755
6.663.1 Detailed Description . . . . .	3755
6.664 stdckdint.h File Reference . . . . .	3755



6.664.1 Detailed Description . . . . .	3755
6.665 stdexcept File Reference . . . . .	3755
6.665.1 Detailed Description . . . . .	3756
6.666 stdlib.h File Reference . . . . .	3756
6.666.1 Detailed Description . . . . .	3756
6.666.2 Function Documentation . . . . .	3756
6.667 stop_token File Reference . . . . .	3756
6.667.1 Detailed Description . . . . .	3756
6.668 streambuf File Reference . . . . .	3756
6.668.1 Detailed Description . . . . .	3757
6.669 debug/string File Reference . . . . .	3757
6.669.1 Detailed Description . . . . .	3759
6.670 experimental/string File Reference . . . . .	3759
6.670.1 Detailed Description . . . . .	3760
6.671 string File Reference . . . . .	3760
6.671.1 Detailed Description . . . . .	3760
6.672 experimental/string_view File Reference . . . . .	3760
6.672.1 Detailed Description . . . . .	3762
6.673 string_view File Reference . . . . .	3762
6.673.1 Detailed Description . . . . .	3763
6.674 syncstream File Reference . . . . .	3763
6.674.1 Detailed Description . . . . .	3763
6.675 experimental/system_error File Reference . . . . .	3763
6.675.1 Detailed Description . . . . .	3764
6.676 system_error File Reference . . . . .	3764
6.676.1 Detailed Description . . . . .	3764
6.677 text_encoding File Reference . . . . .	3764
6.677.1 Detailed Description . . . . .	3764
6.678 tgmth.h File Reference . . . . .	3764
6.678.1 Detailed Description . . . . .	3764
6.679 thread File Reference . . . . .	3765
6.679.1 Detailed Description . . . . .	3765
6.680 bool_set File Reference . . . . .	3765
6.680.1 Detailed Description . . . . .	3766
6.681 bool_set.tcc File Reference . . . . .	3766
6.681.1 Detailed Description . . . . .	3766
6.682 dynamic_bitset File Reference . . . . .	3766
6.682.1 Detailed Description . . . . .	3767
6.683 dynamic_bitset.tcc File Reference . . . . .	3767

6.683.1 Detailed Description . . . . .	3767
6.684 experimental/tuple File Reference . . . . .	3767
6.684.1 Detailed Description . . . . .	3768
6.685 tuple File Reference . . . . .	3768
6.685.1 Detailed Description . . . . .	3770
6.686 experimental/type_traits File Reference . . . . .	3770
6.686.1 Detailed Description . . . . .	3773
6.687 tr2/type_traits File Reference . . . . .	3773
6.687.1 Detailed Description . . . . .	3774
6.688 type_traits File Reference . . . . .	3774
6.688.1 Detailed Description . . . . .	3778
6.689 typeindex File Reference . . . . .	3778
6.689.1 Detailed Description . . . . .	3778
6.690 debug/unordered_map File Reference . . . . .	3778
6.690.1 Detailed Description . . . . .	3780
6.691 experimental/unordered_map File Reference . . . . .	3780
6.691.1 Detailed Description . . . . .	3781
6.692 unordered_map File Reference . . . . .	3781
6.692.1 Detailed Description . . . . .	3781
6.693 debug/unordered_set File Reference . . . . .	3781
6.693.1 Detailed Description . . . . .	3783
6.694 experimental/unordered_set File Reference . . . . .	3783
6.694.1 Detailed Description . . . . .	3784
6.695 unordered_set File Reference . . . . .	3784
6.695.1 Detailed Description . . . . .	3784
6.696 experimental/utility File Reference . . . . .	3785
6.696.1 Detailed Description . . . . .	3785
6.697 utility File Reference . . . . .	3785
6.697.1 Detailed Description . . . . .	3785
6.698 valarray File Reference . . . . .	3785
6.698.1 Detailed Description . . . . .	3786
6.699 variant File Reference . . . . .	3786
6.699.1 Detailed Description . . . . .	3786
6.700 debug/vector File Reference . . . . .	3786
6.700.1 Detailed Description . . . . .	3787
6.701 experimental/vector File Reference . . . . .	3787
6.701.1 Detailed Description . . . . .	3787
6.702 vector File Reference . . . . .	3788
6.702.1 Detailed Description . . . . .	3788

6.703 atomic_word.h File Reference . . . . .	3788
6.703.1 Detailed Description . . . . .	3788
6.704 basic_file.h File Reference . . . . .	3788
6.704.1 Detailed Description . . . . .	3788
6.705 c++allocator.h File Reference . . . . .	3788
6.705.1 Detailed Description . . . . .	3789
6.706 c++config.h File Reference . . . . .	3789
6.706.1 Detailed Description . . . . .	3796
6.707 c++io.h File Reference . . . . .	3796
6.707.1 Detailed Description . . . . .	3796
6.708 c++locale.h File Reference . . . . .	3796
6.708.1 Detailed Description . . . . .	3796
6.709 c++locale_internal.h File Reference . . . . .	3796
6.709.1 Detailed Description . . . . .	3797
6.710 parallel/compatibility.h File Reference . . . . .	3797
6.710.1 Detailed Description . . . . .	3797
6.711 x86_64-pc-linux-gnu/bits/compatibility.h File Reference . . . . .	3797
6.711.1 Detailed Description . . . . .	3797
6.712 cpu_defines.h File Reference . . . . .	3797
6.712.1 Detailed Description . . . . .	3797
6.713 ctype_base.h File Reference . . . . .	3797
6.713.1 Detailed Description . . . . .	3797
6.714 ctype_inline.h File Reference . . . . .	3798
6.714.1 Detailed Description . . . . .	3798
6.715 cxxabi_tweaks.h File Reference . . . . .	3798
6.715.1 Detailed Description . . . . .	3798
6.716 error_constants.h File Reference . . . . .	3798
6.716.1 Detailed Description . . . . .	3799
6.717 extc++.h File Reference . . . . .	3799
6.717.1 Detailed Description . . . . .	3799
6.718 messages_members.h File Reference . . . . .	3799
6.718.1 Detailed Description . . . . .	3799
6.719 opt_random.h File Reference . . . . .	3799
6.719.1 Detailed Description . . . . .	3799
6.720 os_defines.h File Reference . . . . .	3799
6.720.1 Detailed Description . . . . .	3799
6.721 stdc++.h File Reference . . . . .	3800
6.721.1 Detailed Description . . . . .	3800
6.722 stdtr1c++.h File Reference . . . . .	3800

6.722.1 Detailed Description . . . . .	3800
6.723 time_members.h File Reference . . . . .	3800
6.723.1 Detailed Description . . . . .	3800
6.724 compare File Reference . . . . .	3800
6.724.1 Detailed Description . . . . .	3801
6.725 cxxabi.h File Reference . . . . .	3801
6.725.1 Detailed Description . . . . .	3802
6.725.2 Function Documentation . . . . .	3802
6.726 exception File Reference . . . . .	3803
6.726.1 Detailed Description . . . . .	3804
6.727 initializer_list File Reference . . . . .	3804
6.727.1 Detailed Description . . . . .	3804
6.728 new File Reference . . . . .	3804
6.728.1 Detailed Description . . . . .	3805
6.728.2 Function Documentation . . . . .	3805
6.729 typeinfo File Reference . . . . .	3805
6.729.1 Detailed Description . . . . .	3806
<b>Index</b>	<b>3807</b>

## 1 Deprecated List

### Module `negators`

Deprecated in C++17, no longer in the standard since C++20. Use `not_fn` instead.

### Module `pointer_adaptors`

Deprecated in C++11, no longer in the standard since C++17.

### Module `ptrmem_adaptors`

Deprecated in C++11, no longer in the standard since C++17. Use `mem_fn` instead.

### Struct `std::aligned_storage<_Len, _Align>`

Deprecated in C++23. Uses can be replaced by an array `std::byte[_Len]` declared with `alignas` ( $\leftarrow$  `_Align`).

### Struct `std::aligned_union<_Len, _Types>`

Deprecated in C++23.

### Class `std::auto_ptr<_Tp>`

Deprecated in C++11, no longer in the standard since C++17. Use `unique_ptr` instead.

Struct `std::binary_function<_Arg1, _Arg2, _Result >`

Deprecated in C++11, no longer in the standard since C++17.

Member `std::get_unexpected () noexcept`

Removed from the C++ standard in C++17

Struct `std::is_literal_type<_Tp >`

Deprecated in C++17, removed in C++20. The idea of a literal type isn't useful.

Struct `std::is_pod<_Tp >`

Deprecated in C++20. Use `is_standard_layout` && `is_trivial` instead.

Struct `std::is_trivial<_Tp >`

Deprecated in C++26. Use a combination of one or more more specialized type traits instead, such as `is_trivially_default_constructible`, `is_trivially_copy_constructible`, `is_trivially_copy_assignable`, etc., depending on the exact check(s) needed.

Member `std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`

Since C++17, `std::random_shuffle` is not part of the C++ standard. Use `std::shuffle` instead, which was introduced in C++11.

Member `std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`

Since C++17, `std::random_shuffle` is not part of the C++ standard. Use `std::shuffle` instead, which was introduced in C++11.

Member `std::set_unexpected (unexpected_handler) noexcept`

Removed from the C++ standard in C++17

Struct `std::unary_function<_Arg, _Result >`

Deprecated in C++11, no longer in the standard since C++17.

Member `std::unexpected ()`

Removed from the C++ standard in C++17

## 2 Todo List

Member `__gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Class `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc >`

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Class `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>`

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>`

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class `__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc>`

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member `__gnu_cxx::power` (`_Tp __x, _Integer __n, _MonoidOperation __monoid_op`)

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member `__gnu_cxx::power` (`_Tp __x, _Integer __n`)

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member `__gnu_cxx::random_sample` (`_InputIterator __first, _InputIterator __last, _RandomAccessIterator __↵out_first, _RandomAccessIterator __out_last`)

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member `__gnu_cxx::random_sample` (`_InputIterator __first, _InputIterator __last, _RandomAccessIterator __↵out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand`)

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member `__gnu_cxx::random_sample_n` (`_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __↵out, const _Distance __n`)

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member `__gnu_cxx::random_sample_n` (`_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __↵out, const _Distance __n, _RandomNumberGenerator &__rand`)

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Struct `__gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc>`

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class `__gnu_cxx::rope<_CharT, _Alloc>`

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class `__gnu_cxx::slist<_Tp, _Alloc>`

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Module `mathsf`

Provide accuracy comparisons on a per-function basis for a small number of targets.

Class `std::basic_string<_CharT, _Traits, _Alloc>`

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member `std::regex_traits<_Ch_type>::transform_primary(_Fwd_iter __first, _Fwd_iter __last) const`

Implement this function correctly.

## 3 Topic Documentation

### 3.1 Algorithms

Collaboration diagram for Algorithms:



#### Topics

- [Generalized Numeric operations](#)
- [Mutating](#)
- [Non-Mutating](#)
- [Sorting](#)

### 3.1.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

### 3.1.2 Generalized Numeric operations

Collaboration diagram for Generalized Numeric operations:



### Functions

- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __↔  
binary_op)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __↔  
__result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __↔  
__result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator std::exclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↔  
result, _Tp __init)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _OutputIterator std::exclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↔  
result, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↔  
result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↔  
result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>`  
`constexpr _OutputIterator std::inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↔  
result, _BinaryOperation __binary_op, _Tp __init)`



- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`  
`constexpr _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator>`  
`constexpr iterator_traits< _InputIterator >::value_type std::reduce (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator std::transform_exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator std::transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>`  
`constexpr _OutputIterator std::transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _Tp std::transform_reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp std::transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`  
`constexpr _Tp std::transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`

### 3.1.2.1 Detailed Description

### 3.1.2.2 Function Documentation

#### **accumulate()** [1/2]

```
template<typename _InputIterator, typename _Tp>
_Tp std::accumulate (
    _InputIterator __first,
    _InputIterator __last,
    _Tp __init) [inline], [constexpr]
```

Accumulate values in a range.

Accumulates the values in the range `[first,last)` using `operator+()`. The initial value is *init*. The values are processed in order.

#### Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.

#### Returns

The final sum.

#### **accumulate()** [2/2]

```
template<typename _InputIterator, typename _Tp, typename _BinaryOperation>
_Tp std::accumulate (
    _InputIterator __first,
    _InputIterator __last,
    _Tp __init,
    _BinaryOperation __binary_op) [inline], [constexpr]
```

Accumulate values in a range with operation.

Accumulates the values in the range `[first,last)` using the function object `__binary_op`. The initial value is `__init`. The values are processed in order.

#### Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op</code>	Function object to accumulate with.

#### Returns

The final sum.

#### **adjacent\_difference()** [1/2]

```
template<typename _InputIterator, typename _OutputIterator>
_OutputIterator std::adjacent_difference (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result) [constexpr]
```

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[first,last)` using `operator-()` and writes the result to `__result`.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sums.

**Returns**

Iterator pointing just beyond the values written to result.

**adjacent\_difference()** [2/2]

```
template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator std::adjacent_difference (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op) [constexpr]
```

Return differences between adjacent values.

Computes the difference between adjacent values in the range [`__first`,`__last`) using the function object `__binary_op` and writes the result to `__result`.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

**Returns**

Iterator pointing just beyond the values written to result.

**exclusive\_scan()** [1/2]

```
template<typename _InputIterator, typename _OutputIterator, typename _Tp>
_OutputIterator std::exclusive_scan (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Tp __init) [inline], [constexpr]
```

Output the cumulative sum of one range to a second range.

## Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Start of output range.
<code>__init</code>	Initial value.

## Returns

The end of the output range.

Write the cumulative sum (aka prefix sum, aka scan) of the input range to the output range. Each element of the output range contains the running total of all earlier elements (and the initial value), using `std::plus<>` for summation.

This function generates an "exclusive" scan, meaning the Nth element of the output range is the sum of the first N-1 input elements, so the Nth input element is not included.

**exclusive\_scan()** [2/2]

```
template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>
_OutputIterator std::exclusive_scan (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Tp __init,
    _BinaryOperation __binary_op) [constexpr]
```

Output the cumulative sum of one range to a second range.

## Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Start of output range.
<code>__init</code>	Initial value.
<code>__binary_op</code>	Function to perform summation.

## Returns

The end of the output range.

Write the cumulative sum (aka prefix sum, aka scan) of the input range to the output range. Each element of the output range contains the running total of all earlier elements (and the initial value), using `binary_op` for summation.

This function generates an "exclusive" scan, meaning the Nth element of the output range is the sum of the first N-1 input elements, so the Nth input element is not included.

**inclusive\_scan()** [1/3]

```
template<typename _InputIterator, typename _OutputIterator>
_OutputIterator std::inclusive_scan (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result) [inline], [constexpr]
```

Output the cumulative sum of one range to a second range.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Start of output range.

**Returns**

The end of the output range.

Write the cumulative sum (aka prefix sum, aka scan) of the input range to the output range. Each element of the output range contains the running total of all earlier elements, using `std::plus<>` for summation.

This function generates an "inclusive" scan, meaning the Nth element of the output range is the sum of the first N input elements, so the Nth input element is included.

**inclusive\_scan()** [2/3]

```
template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator std::inclusive_scan (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op) [constexpr]
```

Output the cumulative sum of one range to a second range.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Start of output range.
<code>__binary_op</code>	Function to perform summation.

**Returns**

The end of the output range.

Write the cumulative sum (aka prefix sum, aka scan) of the input range to the output range. Each element of the output range contains the running total of all earlier elements, using `binary_op` for summation.

This function generates an "inclusive" scan, meaning the Nth element of the output range is the sum of the first N input elements, so the Nth input element is included.

**inclusive\_scan()** [3/3]

```
template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename ↵
_Tp>
_OutputIterator std::inclusive_scan (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op,
    _Tp __init) [constexpr]
```

Output the cumulative sum of one range to a second range.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Start of output range.
<code>__binary_op</code>	Function to perform summation.
<code>__init</code>	Initial value.

**Returns**

The end of the output range.

Write the cumulative sum (aka prefix sum, aka scan) of the input range to the output range. Each element of the output range contains the running total of all earlier elements (and the initial value), using `binary_op` for summation.

This function generates an "inclusive" scan, meaning the Nth element of the output range is the sum of the first N input elements, so the Nth input element is included.

**inner\_product()** [1/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _Tp>
_Tp std::inner_product (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _Tp __init) [inline], [constexpr]
```

Compute inner product of two ranges.

Starting with an initial value of `__init`, multiplies successive elements from the two ranges and adds each product into the accumulated value using `operator+()`. The values in the ranges are processed in order.

**Parameters**

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.

**Returns**

The final inner product.

**inner\_product()** [2/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _Binary↵
Operation1, typename _BinaryOperation2>
_Tp std::inner_product (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _Tp __init,
    _BinaryOperation1 __binary_op1,
    _BinaryOperation2 __binary_op2) [inline], [constexpr]
```

Compute inner product of two ranges.

Starting with an initial value of `__init`, applies `__binary_op2` to successive elements from the two ranges and accumulates each result into the accumulated value using `__binary_op1`. The values in the ranges are processed in order.

**Parameters**

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op1</code>	Function object to accumulate with.
<code>__binary_op2</code>	Function object to apply to pairs of input values.

**Returns**

The final inner product.

**iota()**

```
template<typename _ForwardIterator, typename _Tp>
void std::iota (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Tp __value) [constexpr]
```

Create a range of sequentially increasing values.

For each element in the range `[first,last)` assigns `value` and increments `value` as if by `++value`.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__value</code>	Starting value.

**Returns**

Nothing.

**partial\_sum()** [1/2]

```
template<typename _InputIterator, typename _OutputIterator>
_OutputIterator std::partial_sum (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result)  [constexpr]
```

Return list of partial sums.

Accumulates the values in the range [first,last) using the + operator. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.

**Returns**

Iterator pointing just beyond the values written to `__result`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_sequential\\_random\\_shuffle\(\)](#).

**partial\_sum()** [2/2]

```
template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator std::partial_sum (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op)  [constexpr]
```

Return list of partial sums.

Accumulates the values in the range [first,last) using `__binary_op`. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

**Returns**

Iterator pointing just beyond the values written to `__result`.



**reduce()** [1/3]

```
template<typename _InputIterator>
iterator_traits< _InputIterator >::value_type std::reduce (
    _InputIterator __first,
    _InputIterator __last) [inline], [constexpr]
```

Calculate reduction of values in a range.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

The final sum.

Reduce the values in the range `[first,last)` using addition, with an initial value of `T{}`, where `T` is the iterator's value type. Equivalent to calling `std::reduce(first, last, T{}, std::plus<>())`.

**reduce()** [2/3]

```
template<typename _InputIterator, typename _Tp>
_Tp std::reduce (
    _InputIterator __first,
    _InputIterator __last,
    _Tp __init) [inline], [constexpr]
```

Calculate reduction of values in a range.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.

**Returns**

The final sum.

Reduce the values in the range `[first,last)` using addition. Equivalent to calling `std::reduce(first, last, init, std::plus<>())`.

**reduce()** [3/3]

```
template<typename _InputIterator, typename _Tp, typename _BinaryOperation>
_Tp std::reduce (
    _InputIterator __first,
    _InputIterator __last,
    _Tp __init,
    _BinaryOperation __binary_op) [constexpr]
```

Calculate reduction of values in a range.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op</code>	A binary function object.

## Returns

The final sum.

Reduce the values in the range `[first, last)` using a binary operation. The initial value is `init`. The values are not necessarily processed in order.

This algorithm is similar to `std::accumulate` but is not required to perform the operations in order from first to last. For operations that are commutative and associative the result will be the same as for `std::accumulate`, but for other operations (such as floating point arithmetic) the result can be different.

**transform\_exclusive\_scan()**

```
template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _Binary←
Operation, typename _UnaryOperation>
_OutputIterator std::transform_exclusive_scan (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Tp __init,
    _BinaryOperation __binary_op,
    _UnaryOperation __unary_op) [constexpr]
```

Output the cumulative sum of one range to a second range.

## Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Start of output range.
<code>__init</code>	Initial value.
<code>__binary_op</code>	Function to perform summation.
<code>__unary_op</code>	Function to transform elements of the input range.

## Returns

The end of the output range.

Write the cumulative sum (aka prefix sum, aka scan) of the input range to the output range. Each element of the output range contains the running total of all earlier elements (and the initial value), using `__unary_op` to transform the input elements and using `__binary_op` for summation.

This function generates an "exclusive" scan, meaning the Nth element of the output range is the sum of the first N-1 input elements, so the Nth input element is not included.

**transform\_inclusive\_scan()** [1/2]

```
template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>
_OutputIterator std::transform_inclusive_scan (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op,
    _UnaryOperation __unary_op) [constexpr]
```

Output the cumulative sum of one range to a second range.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Start of output range.
<code>__binary_op</code>	Function to perform summation.
<code>__unary_op</code>	Function to transform elements of the input range.

**Returns**

The end of the output range.

Write the cumulative sum (aka prefix sum, aka scan) of the input range to the output range. Each element of the output range contains the running total of all earlier elements, using `__unary_op` to transform the input elements and using `__binary_op` for summation.

This function generates an "inclusive" scan, meaning the Nth element of the output range is the sum of the first N input elements, so the Nth input element is included.

**transform\_inclusive\_scan()** [2/2]

```
template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>
_OutputIterator std::transform_inclusive_scan (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op,
    _UnaryOperation __unary_op,
    _Tp __init) [constexpr]
```

Output the cumulative sum of one range to a second range.

**Parameters**

<code>__first</code>	Start of input range.
----------------------	-----------------------

<code>__last</code>	End of input range.
<code>__result</code>	Start of output range.
<code>__binary_op</code>	Function to perform summation.
<code>__unary_op</code>	Function to transform elements of the input range.
<code>__init</code>	Initial value.

#### Returns

The end of the output range.

Write the cumulative sum (aka prefix sum, aka scan) of the input range to the output range. Each element of the output range contains the running total of all earlier elements (and the initial value), using `__unary_op` to transform the input elements and using `__binary_op` for summation.

This function generates an "inclusive" scan, meaning the Nth element of the output range is the sum of the first N input elements, so the Nth input element is included.

#### **transform\_reduce()** [1/3]

```
template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _Unary↵
Operation>
_Tp std::transform_reduce (
    _InputIterator __first,
    _InputIterator __last,
    _Tp __init,
    _BinaryOperation __binary_op,
    _UnaryOperation __unary_op) [constexpr]
```

Transform the elements of a range and reduce.

#### Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op</code>	The function used to perform reduction.
<code>__unary_op</code>	The function used to transform values from the range.

#### Returns

The final sum.

Call `unary_op(first[n])` for each `n` in `[0, last-first)` and then use `binary_op` to reduce the values returned by `unary_op` to a single value of type `T`.

**transform\_reduce()** [2/3]

```
template<typename _InputIterator1, typename _InputIterator2, typename _Tp>
_Tp std::transform_reduce (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _Tp __init) [inline], [constexpr]
```

Combine elements from two ranges and reduce.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__init</code>	Starting value to add other values to.

**Returns**

The final sum.

Call `first1[n]*first2[n]` for each `n` in `[0, last1-first1)` and then use addition to sum those products to a single value of type `T`.

The range beginning at `first2` must contain at least `last1-first1` elements.

**transform\_reduce()** [3/3]

```
template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOp1,
typename _BinaryOperation2>
_Tp std::transform_reduce (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _Tp __init,
    _BinaryOperation1 __binary_op1,
    _BinaryOperation2 __binary_op2) [constexpr]
```

Combine elements from two ranges and reduce.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op1</code>	The function used to perform reduction.
<code>__binary_op2</code>	The function used to combine values from the ranges.

**Returns**

The final sum.

Call `binary_op2(first1[n], first2[n])` for each `n` in `[0, last1-first1)` and then use `binary_op1` to reduce the values returned by `binary_op2` to a single value of type `T`.

The range beginning at `first2` must contain at least `last1-first1` elements.

**3.1.3 Mutating**

Collaboration diagram for Mutating:

**Functions**

- `template<typename _II, typename _OI>`  
`constexpr _OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2>`  
`constexpr _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _↵`  
`Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`constexpr _OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp>`  
`constexpr _OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator>`  
`constexpr void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`constexpr _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI>`  
`constexpr _OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2>`  
`constexpr _BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`

- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>`  
`constexpr pair< _OutputIterator1, _OutputIterator2 > std::partition\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::partition\_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator>`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator std::remove\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>`  
`constexpr _OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp>`  
`constexpr void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator>`  
`constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>`  
`constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵`  
`binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`_BinaryPredicate __binary_pred)`

### 3.1.3.1 Detailed Description

### 3.1.3.2 Function Documentation

#### **copy()**

```
template<typename _II, typename _OI>
_OI std::copy (
    _II __first,
    _II __last,
    _OI __result) [inline], [constexpr]
```

Copies the range [first,last) into result.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

#### Returns

`result + (last - first)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

#### **copy\_backward()**

```
template<typename _BI1, typename _BI2>
_BI2 std::copy_backward (
    _BI1 __first,
    _BI1 __last,
    _BI2 __result) [inline], [constexpr]
```

Copies the range [first,last) into result.



**Parameters**

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

**Returns**

result - (last - first)

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `[first,last]`. Use `copy` instead. Note that the start of the output range may overlap `[first,last]`.

**copy\_if()**

```
template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
_OutputIterator std::copy_if (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Predicate __pred) [constexpr]
```

Copy the elements of a sequence for which a predicate is true.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

**Returns**

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns true to the range beginning at `__result`.

`copy_if()` is stable, so the relative order of elements that are copied is unchanged.

**copy\_n()**

```
template<typename _InputIterator, typename _Size, typename _OutputIterator>
_OutputIterator std::copy_n (
    _InputIterator __first,
    _Size __n,
    _OutputIterator __result) [inline], [constexpr]
```

Copies the range `[first,first+n)` into `[result,result+n)`.

## Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

## Returns

`result+n`.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

References [move\(\)](#).

**fill()**

```
template<typename _ForwardIterator, typename _Tp>
void std::fill (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __value) [inline], [constexpr]
```

Fills the range `[first,last)` with copies of `value`.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__value</code>	A reference-to-const of arbitrary type.

## Returns

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

**fill\_n()**

```
template<typename _OI, typename _Size, typename _Tp>
_OI std::fill_n (
    _OI __first,
    _Size __n,
    const _Tp & __value) [inline], [constexpr]
```

Fills the range `[first,first+n)` with copies of `value`.

**Parameters**

<code>__first</code>	An output iterator.
<code>__n</code>	The count of copies to perform.
<code>__value</code>	A reference-to-const of arbitrary type.

**Returns**

The iterator at `first+n`.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

If `__n` is negative, the function does nothing.

References [\\_\\_iterator\\_category\(\)](#).

**generate()**

```
template<typename _ForwardIterator, typename _Generator>
void std::generate (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Generator __gen) [constexpr]
```

Assign the result of a function object to each value in a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__gen</code>	A function object callable with no arguments.

**Returns**

`generate()` returns no value.

Performs the assignment `*i = __gen()` for each `i` in the range `[__first, __last)`.

**generate\_n()**

```
template<typename _OutputIterator, typename _Size, typename _Generator>
_OutputIterator std::generate_n (
    _OutputIterator __first,
    _Size __n,
    _Generator __gen) [constexpr]
```

Assign the result of a function object to each value in a sequence.

## Parameters

<code>__first</code>	A forward iterator.
<code>__n</code>	The length of the sequence.
<code>__gen</code>	A function object callable with no arguments.

## Returns

The end of the sequence, i.e., `__first + __n`

Performs the assignment `*i = __gen()` for each `i` in the range `[__first, __first + __n)`.

If `__n` is negative, the function does nothing and returns `__first`.

**is\_partitioned()**

```
template<typename _InputIterator, typename _Predicate>
bool std::is_partitioned (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred) [inline], [nodiscard], [constexpr]
```

Checks whether the sequence is partitioned.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

True if the range `[__first,__last)` is partitioned by `__pred`, i.e. if all elements that satisfy `__pred` appear before those that do not.

**iter\_swap()**

```
template<typename _ForwardIterator1, typename _ForwardIterator2>
void std::iter_swap (
    _ForwardIterator1 __a,
    _ForwardIterator2 __b) [inline], [constexpr]
```

Swaps the contents of two iterators.

**Parameters**

$\_a \leftrightarrow$	An iterator.
$\_b \leftrightarrow$	Another iterator.

**Returns**

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

**move()**

```
template<typename _II, typename _OI>
_OI std::move (
    _II __first,
    _II __last,
    _OI __result) [inline], [constexpr]
```

Moves the range [first,last) into result.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

**Returns**

result + (last - first)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

**move\_backward()**

```
template<typename _BI1, typename _BI2>
_BI2 std::move_backward (
    _BI1 __first,
    _BI1 __last,
    _BI2 __result) [inline], [constexpr]
```

Moves the range [first,last) into result.

**Parameters**

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

**Returns**

`result - (last - first)`

The function has the same effect as `move`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `[first,last]`. Use `move` instead. Note that the start of the output range may overlap `[first,last]`.

**partition()**

```
template<typename _ForwardIterator, typename _Predicate>
_FowardIterator std::partition (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred) [inline], [constexpr]
```

Move elements for which a predicate is true to the beginning of a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

**Returns**

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[__first, middle)` and false for each `i` in the range `[middle, __last)`.

`__pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

References [\\_\\_iterator\\_category\(\)](#), and [\\_\\_partition\(\)](#).

**partition\_copy()**

```
template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename
_Predicate>
pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator1 __out_true,
    _OutputIterator2 __out_false,
    _Predicate __pred) [constexpr]
```

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__out_true</code>	An output iterator.
<code>__out_false</code>	An output iterator.
<code>__pred</code>	A predicate.

## Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range `[__first, __last)` for which `__pred` returns true to the range beginning at `__out_true` and each element for which `__pred` returns false to `__out_false`.

**partition\_point()**

```
template<typename _ForwardIterator, typename _Predicate>
_FowardIterator std::partition_point (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred) [nodiscard], [constexpr]
```

Find the partition point of a partitioned range.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__pred</code>	A predicate.

## Returns

An iterator `mid` such that `all_of(__first, mid, __pred)` and `none_of(mid, __last, __pred)` are both true.

References [advance\(\)](#), and [distance\(\)](#).

**random\_shuffle()** [1/2]

```
template<typename _RandomAccessIterator>
void std::random_shuffle (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last) [inline]
```

Randomly shuffle the elements of a sequence.



**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Returns**

Nothing.

Reorder the elements in the range `[__first, __last)` using a random distribution, so that every possible ordering of the sequence is equally likely.

**Deprecated** Since C++17, `std::random_shuffle` is not part of the C++ standard. Use `std::shuffle` instead, which was introduced in C++11.

**random\_shuffle()** [2/2]

```
template<typename _RandomAccessIterator, typename _RandomNumberGenerator>
void std::random_shuffle (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _RandomNumberGenerator && __rand)
```

Shuffle the elements of a sequence using a random number generator.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__rand</code>	The RNG functor or function.

**Returns**

Nothing.

Reorders the elements in the range `[__first, __last)` using `__rand` to provide a random distribution. Calling `__rand(N)` for a positive integer `N` should return a randomly chosen integer from the range `[0, N)`.

**Deprecated** Since C++17, `std::random_shuffle` is not part of the C++ standard. Use `std::shuffle` instead, which was introduced in C++11.

**remove()**

```
template<typename _ForwardIterator, typename _Tp>
_FowardIterator std::remove (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __value) [inline], [nodiscard], [constexpr]
```

Remove elements from a sequence.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be removed.

**Returns**

An iterator designating the end of the resulting sequence.

All elements equal to `__value` are removed from the range `[__first,__last)`.

`remove()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

**remove\_copy()**

```
template<typename _InputIterator, typename _OutputIterator, typename _Tp>
_OutputIterator std::remove_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    const _Tp & __value) [inline], [constexpr]
```

Copy a sequence, removing elements of a given value.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__value</code>	The value to be removed.

**Returns**

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` not equal to `__value` to the range beginning at `__result`.

`remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

**remove\_copy\_if()**

```
template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
_OutputIterator std::remove_copy_if (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Predicate __pred) [inline], [constexpr]
```

Copy a sequence, removing elements for which a predicate is true.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

**Returns**

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns false to the range beginning at `__result`.

`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

**remove\_if()**

```
template<typename _ForwardIterator, typename _Predicate>
_FowardIterator std::remove_if (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred) [inline], [nodiscard], [constexpr]
```

Remove elements from a sequence using a predicate.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.

**Returns**

An iterator designating the end of the resulting sequence.

All elements for which `__pred` returns true are removed from the range `[__first,__last)`.

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

**replace()**

```
template<typename _ForwardIterator, typename _Tp>
void std::replace (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __old_value,
    const _Tp & __new_value) [constexpr]
```

Replace each occurrence of one value in a sequence with another value.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

## Returns

`replace()` returns no value.

For each iterator `i` in the range `[__first, __last)` if `*i == __old_value` then the assignment `*i = __new_value` is performed.

**`replace_copy_if()`**

```
template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>
_OutputIterator std::replace_copy_if (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Predicate __pred,
    const _Tp & __new_value) [inline], [constexpr]
```

Copy a sequence, replacing each value for which a predicate returns true with another value.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

## Returns

The end of the output sequence, `__result+(__last-__first)`.

Copies each element in the range `[__first, __last)` to the range `[__result, __result+(__last-__first))` replacing elements for which `__pred` returns true with `__new_value`.

**`replace_if()`**

```
template<typename _ForwardIterator, typename _Predicate, typename _Tp>
void std::replace_if (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred,
    const _Tp & __new_value) [constexpr]
```

Replace each value in a sequence for which a predicate returns true with another value.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

**Returns**

`replace_if()` returns no value.

For each iterator `i` in the range `[__first, __last)` if `__pred(*i)` is true then the assignment `*i = __new_value` is performed.

**reverse()**

```
template<typename _BidirectionalIterator>
void std::reverse (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last) [inline], [constexpr]
```

Reverse a sequence.

**Parameters**

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.

**Returns**

`reverse()` returns no value.

Reverses the order of the elements in the range `[__first, __last)`, so that the first element becomes the last etc. For every `i` such that  $0 \leq i < (\_\text{last} - \_\text{first})/2$ , `reverse()` swaps `*(__first+i)` and `*(__last-(i+1))`

References [\\_\\_iterator\\_category\(\)](#), and [\\_\\_reverse\(\)](#).

**reverse\_copy()**

```
template<typename _BidirectionalIterator, typename _OutputIterator>
_OutputIterator std::reverse_copy (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _OutputIterator __result) [constexpr]
```

Copy a sequence, reversing its elements.

## Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	An output iterator.

## Returns

An iterator designating the end of the resulting sequence.

Copies the elements in the range `[__first, __last)` to the range `[__result, __result+(__last-__first))` such that the order of the elements is reversed. For every `i` such that  $0 \leq i < (__last - __first)$ , `reverse_copy()` performs the assignment `*(__result+(__last-__first)-1-i) = *(__first+i)`. The ranges `[__first, __last)` and `[__result, __result+(__last-__first))` must not overlap.

**rotate()**

```
template<typename _ForwardIterator>
_FowardIterator std::rotate (
    _ForwardIterator __first,
    _ForwardIterator __middle,
    _ForwardIterator __last) [inline], [constexpr]
```

Rotate the elements of a sequence.

## Parameters

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.

## Returns

`first + (last - middle)`.

Rotates the elements of the range `[__first, __last)` by `(__middle - __first)` positions so that the element at `__middle` is moved to `__first`, the element at `__middle+1` is moved to `__first+1` and so on for each element in the range `[__first, __last)`.

This effectively swaps the ranges `[__first, __middle)` and `[__middle, __last)`.

Performs `*(__first+(n+(__last-__middle))%(__last-__first))=*(__first+n)` for each `n` in the range `[0, __last-__first)`.

References [\\_\\_iterator\\_category\(\)](#), and [\\_\\_rotate\(\)](#).

**rotate\_copy()**

```
template<typename _ForwardIterator, typename _OutputIterator>
_OutputIterator std::rotate_copy (
    _ForwardIterator __first,
    _ForwardIterator __middle,
    _ForwardIterator __last,
    _OutputIterator __result) [inline], [constexpr]
```

Copy a sequence, rotating its elements.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__result</code>	An output iterator.

**Returns**

An iterator designating the end of the resulting sequence.

Copies the elements of the range `[__first,__last)` to the range beginning at

**Returns**

, rotating the copied elements by `(__middle-__first)` positions so that the element at `__middle` is moved to `__result`, the element at `__middle+1` is moved to `__result+1` and so on for each element in the range `[__first,__last)`.

Performs `*(__result+(n+(__last-__middle))%(__last-__first))=*(__first+n)` for each `n` in the range `[0,__last-__first)`.

**shuffle()**

```
template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>
void std::shuffle (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _UniformRandomNumberGenerator && __g)
```

Shuffle the elements of a sequence using a uniform random number generator.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__g</code>	A <code>UniformRandomNumberGenerator</code> (26.5.1.3).

**Returns**

Nothing.

Reorders the elements in the range `[__first,__last)` using `__g` to provide random numbers.

References [\\_\\_gen\\_two\\_uniform\\_ints\(\)](#), [std::pair<\\_T1,\\_T2>::first](#), and [std::pair<\\_T1,\\_T2>::second](#).

**stable\_partition()**

```
template<typename _ForwardIterator, typename _Predicate>
_FowardIterator std::stable_partition (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred) [inline]
```

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

**Returns**

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[first,middle)` and false for each `i` in the range `[middle,last)`.

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `__pred(x) == __pred(y)` will have the same relative ordering after calling `stable_partition()`.

**swap\_ranges()**

```
template<typename _ForwardIterator1, typename _ForwardIterator2>
_FForwardIterator2 std::swap_ranges (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2) [constexpr]
```

Swap the elements of two sequences.

**Parameters**

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.

**Returns**

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

**transform() [1/2]**

```
template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>
_OutputIterator std::transform (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _UnaryOperation __unary_op) [constexpr]
```

Perform an operation on a sequence.



**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__unary_op</code>	A unary operator.

**Returns**

An output iterator equal to `__result+(__last-__first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=unary_op(*(__first+N))` for each `N` in the range `[0,__last-__first)`.

`unary_op` must not alter its argument.

**transform() [2/2]**

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename ↵
_BinaryOperation>
_OutputIterator std::transform (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _OutputIterator __result,
    _BinaryOperation __binary_op) [constexpr]
```

Perform an operation on corresponding elements of two sequences.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_op</code>	A binary operator.

**Returns**

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `(__result+N)=__binary_op(*(__first1+N), (__first2+N))` for each `N` in the range `[0,__last1-__first1)`.

`binary_op` must not alter either of its arguments.

**unique()** [1/2]

```
template<typename _ForwardIterator>
_FForwardIterator std::unique (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [nodiscard], [constexpr]
```

Remove consecutive duplicate values from a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Returns**

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

**unique() [2/2]**

```
template<typename _ForwardIterator, typename _BinaryPredicate>
_FForwardIterator std::unique (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _BinaryPredicate __binary_pred) [inline], [nodiscard], [constexpr]
```

Remove consecutive values from a sequence using a predicate.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

**Returns**

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `__binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

**unique\_copy() [1/2]**

```
template<typename _InputIterator, typename _OutputIterator>
_OutputIterator std::unique_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result) [inline], [constexpr]
```

Copy a sequence, removing consecutive duplicate values.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

## Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first, __last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

References [\\_\\_iterator\\_category\(\)](#).

**unique\_copy()** [2/2]

```
template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
_OutputIterator std::unique_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryPredicate __binary_pred) [inline], [constexpr]
```

Copy a sequence, removing consecutive values using a predicate.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_pred</code>	A binary predicate.

## Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first, __last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements for which `__binary_pred` returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

References [\\_\\_iterator\\_category\(\)](#).

### 3.1.4 Non-Mutating

Collaboration diagram for Non-Mutating:



### Functions

- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp & __value)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::count\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _I1, typename _I2>`  
`constexpr bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _I1, typename _I2>`  
`constexpr bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _BinaryPredicate>`  
`constexpr bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2, typename _BinaryPredicate>`  
`constexpr bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`

- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function>`  
`constexpr _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _InputIterator, typename _Size, typename _Function>`  
`constexpr _InputIterator std::for_each_n (_InputIterator __first, _Size __n, _Function __f)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp>`  
`constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

### 3.1.4.1 Detailed Description

### 3.1.4.2 Function Documentation

#### **adjacent\_find()** [1/2]

```
template<typename _ForwardIterator>
_FForwardIterator std::adjacent_find (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [nodiscard], [constexpr]
```

Find two adjacent values in a sequence that are equal.

#### Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

#### Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first,__last)` and such that `*i == *(i+1)`, or `__last` if no such iterator exists.

#### **adjacent\_find()** [2/2]

```
template<typename _ForwardIterator, typename _BinaryPredicate>
_FForwardIterator std::adjacent_find (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _BinaryPredicate __binary_pred) [inline], [nodiscard], [constexpr]
```

Find two adjacent values in a sequence using a predicate.

#### Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

#### Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first,__last)` and such that `__binary_pred(*i,*(i+1))` is true, or `__last` if no such iterator exists.

#### **all\_of()**

```
template<typename _InputIterator, typename _Predicate>
bool std::all_of (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred) [inline], [nodiscard], [constexpr]
```

Checks that a predicate is true for all the elements of a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

True if the check is true, false otherwise.

Returns true if `__pred` is true for each element in the range `[__first,__last)`, and false otherwise.

**any\_of()**

```
template<typename _InputIterator, typename _Predicate>
bool std::any_of (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred) [inline], [nodiscard], [constexpr]
```

Checks that a predicate is true for at least one element of a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

True if the check is true, false otherwise.

Returns true if an element exists in the range `[__first,__last)` such that `__pred` is true, and false otherwise.

**count()**

```
template<typename _InputIterator, typename _Tp>
iterator_traits< _InputIterator >::difference_type std::count (
    _InputIterator __first,
    _InputIterator __last,
    const _Tp & __value) [inline], [nodiscard], [constexpr]
```

Count the number of copies of a value in a sequence.



**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be counted.

**Returns**

The number of iterators `i` in the range `[__first,__last)` for which `*i == __value`

**count\_if()**

```
template<typename _InputIterator, typename _Predicate>
iterator_traits< _InputIterator >::difference_type std::count_if (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred) [inline], [nodiscard], [constexpr]
```

Count the elements of a sequence for which a predicate is true.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

**Returns**

The number of iterators `i` in the range `[__first,__last)` for which `__pred(*i)` is true.

**equal() [1/4]**

```
template<typename _II1, typename _II2>
bool std::equal (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2) [inline], [nodiscard], [constexpr]
```

Tests a range for element-wise equality.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

**Returns**

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

**equal()** [2/4]

```
template<typename _II1, typename _II2>
bool std::equal (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2,
    _II2 __last2) [inline], [nodiscard], [constexpr]
```

Tests a range for element-wise equality.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

**Returns**

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

**equal()** [3/4]

```
template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
bool std::equal (
    _IIter1 __first1,
    _IIter1 __last1,
    _IIter2 __first2,
    _BinaryPredicate __binary_pred) [inline], [nodiscard], [constexpr]
```

Tests a range for element-wise equality.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

**Returns**

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

**equal()** [4/4]

```
template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
bool std::equal (
    _IIter1 __first1,
    _IIter1 __last1,
    _IIter2 __first2,
    _IIter2 __last2,
    _BinaryPredicate __binary_pred) [inline], [nodiscard], [constexpr]
```

Tests a range for element-wise equality.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

**Returns**

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

**find()**

```
template<typename _InputIterator, typename _Tp>
_InputIterator std::find (
    _InputIterator __first,
    _InputIterator __last,
    const _Tp & __val) [inline], [nodiscard], [constexpr]
```

Find the first occurrence of a value in a sequence.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__val</code>	The value to find.

**Returns**

The first iterator `i` in the range `[__first, __last)` such that `*i == __val`, or `__last` if no such iterator exists.

References [to\\_address\(\)](#).

**find\_end()** [1/2]

```
template<typename _ForwardIterator1, typename _ForwardIterator2>
_FForwardIterator1 std::find_end (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2) [inline], [nodiscard], [constexpr]
```

Find last matching subsequence in a sequence.

**Parameters**

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.

**Returns**

The last iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` and returns an iterator to the `__first` element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first1, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1 - (__last2 - __first2)` where `__last2 - __first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`.

References [\\_\\_iterator\\_category\(\)](#).

**find\_end()** [2/2]

```
template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
_FForwardIterator1 std::find_end (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2,
    _BinaryPredicate __comp) [inline], [nodiscard], [constexpr]
```

Find last matching subsequence in a sequence using a predicate.

**Parameters**

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.
<code>__comp</code>	The predicate to use.

**Returns**

The last iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `predicate(*(i+N), (__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first1, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1 - (__last2 - __first2)` where `__last2 - __first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`.

References [\\_\\_iterator\\_category\(\)](#).

**find\_first\_of() [1/2]**

```
template<typename _InputIterator, typename _ForwardIterator>
_InputIterator std::find_first_of (
    _InputIterator __first1,
    _InputIterator __last1,
    _ForwardIterator __first2,
    _ForwardIterator __last2) [nodiscard], [constexpr]
```

Find element from a set in a sequence.

**Parameters**

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.

**Returns**

The first iterator `i` in the range `[__first1, __last1)` such that `*i == *(i2)` such that `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

**find\_first\_of() [2/2]**

```
template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>
_InputIterator std::find_first_of (
    _InputIterator __first1,
    _InputIterator __last1,
    _ForwardIterator __first2,
    _ForwardIterator __last2,
    _BinaryPredicate __comp) [nodiscard], [constexpr]
```

Find element from a set in a sequence using a predicate.

## Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.
<code>__comp</code>	Predicate to use.

## Returns

The first iterator `i` in the range `[__first1, __last1)` such that `comp(*i, *(i2))` is true and `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

**find\_if()**

```
template<typename _InputIterator, typename _Predicate>
_InputIterator std::find_if (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred) [inline], [nodiscard], [constexpr]
```

Find the first element in a sequence for which a predicate is true.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

The first iterator `i` in the range `[__first, __last)` such that `__pred(*i)` is true, or `__last` if no such iterator exists.

**find\_if\_not()**

```
template<typename _InputIterator, typename _Predicate>
_InputIterator std::find_if_not (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred) [inline], [nodiscard], [constexpr]
```

Find the first element in a sequence for which a predicate is false.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

**Returns**

The first iterator `i` in the range `[__first,__last)` such that `__pred(*i)` is false, or `__last` if no such iterator exists.

References [\\_\\_find\\_if\\_not\(\)](#).

**for\_each()**

```
template<typename _InputIterator, typename _Function>
_Function std::for_each (
    _InputIterator __first,
    _InputIterator __last,
    _Function __f)    [constexpr]
```

Apply a function to every element of a sequence.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__f</code>	A unary function object.

**Returns**

`__f`

Applies the function object `__f` to each element in the range `[first,last)`. `__f` must not modify the order of the sequence. If `__f` has a return value it is ignored.

**for\_each\_n()**

```
template<typename _InputIterator, typename _Size, typename _Function>
_InputIterator std::for_each_n (
    _InputIterator __first,
    _Size __n,
    _Function __f)    [constexpr]
```

Apply a function to every element of a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	A value convertible to an integer.
<code>__f</code>	A unary function object.

## Returns

`__first+__n`

Applies the function object `__f` to each element in the range `[first, first+n)`. `__f` must not modify the order of the sequence. If `__f` has a return value it is ignored.

References [move\(\)](#).

**is\_permutation()** [1/4]

```
template<typename _ForwardIterator1, typename _ForwardIterator2>
bool std::is_permutation (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2) [inline], [constexpr]
```

Checks whether a permutation of the second sequence is equal to the first sequence.

## Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.

## Returns

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, begin)` returns true; otherwise, returns false.

**is\_permutation()** [2/4]

```
template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
bool std::is_permutation (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _BinaryPredicate __pred) [inline], [nodiscard], [constexpr]
```

Checks whether a permutation of the second sequence is equal to the first sequence.



**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__pred</code>	A binary predicate.

**Returns**

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` `begin`, such that `equal(__first1, __last1, __begin, __pred)` returns true; otherwise, returns false.

**is\_permutation() [3/4]**

```
template<typename _ForwardIterator1, typename _ForwardIterator2>
bool std::is_permutation (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2) [inline], [nodiscard], [constexpr]
```

Checks whether a permutaion of the second sequence is equal to the first sequence.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of first range.

**Returns**

true if there exists a permutation of the elements in the range `[__first2, __last2)`, beginning with `ForwardIterator2` `begin`, such that `equal(__first1, __last1, begin)` returns true; otherwise, returns false.

**is\_permutation() [4/4]**

```
template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
bool std::is_permutation (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2,
    _BinaryPredicate __pred) [inline], [nodiscard], [constexpr]
```

Checks whether a permutation of the second sequence is equal to the first sequence.

## Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of first range.
<code>__pred</code>	A binary predicate.

## Returns

true if there exists a permutation of the elements in the range `[__first2, __last2)`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, __begin, __pred)` returns true; otherwise, returns false.

**mismatch()** [1/4]

```
template<typename _InputIterator1, typename _InputIterator2>
pair< _InputIterator1, _InputIterator2 > std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2) [inline], [nodiscard], [constexpr]
```

Finds the places in ranges which don't match.

## Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

## Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using `==` and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

**mismatch()** [2/4]

```
template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
pair< _InputIterator1, _InputIterator2 > std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _BinaryPredicate __binary_pred) [inline], [nodiscard], [constexpr]
```

Finds the places in ranges which don't match.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

**Returns**

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the `binary_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

**mismatch() [3/4]**

```
template<typename _InputIterator1, typename _InputIterator2>
pair< _InputIterator1, _InputIterator2 > std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2) [inline], [nodiscard], [constexpr]
```

Finds the places in ranges which don't match.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

**Returns**

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using `==` and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

**mismatch() [4/4]**

```
template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
pair< _InputIterator1, _InputIterator2 > std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _BinaryPredicate __binary_pred) [inline], [nodiscard], [constexpr]
```

Finds the places in ranges which don't match.

## Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

## Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the `binary_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

**none\_of()**

```
template<typename _InputIterator, typename _Predicate>
bool std::none_of (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred) [inline], [nodiscard], [constexpr]
```

Checks that a predicate is false for all the elements of a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

True if the check is true, false otherwise.

Returns true if `__pred` is false for each element in the range `[__first,__last)`, and false otherwise.

**search()** [1/2]

```
template<typename _ForwardIterator1, typename _ForwardIterator2>
_FForwardIterator1 std::search (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2) [inline], [nodiscard], [constexpr]
```

Search a sequence for a matching sub-sequence.

**Parameters**

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.

**Returns**

The first iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1 - (__last2 - __first2)` where `__last2 - __first2` is the length of the sub-sequence.

This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`

**search() [2/2]**

```
template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
_FowardIterator1 std::search (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2,
    _BinaryPredicate __predicate) [inline], [constexpr]
```

Search a sequence for a matching sub-sequence using a predicate.

**Parameters**

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.
<code>__predicate</code>	A binary predicate.

**Returns**

The first iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `__predicate(*(i+N), *(__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)`, using `__predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `__last1` if no such iterator exists.

**See also**

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

**search\_n()** [1/2]

```
template<typename _ForwardIterator, typename _Integer, typename _Tp>
_FForwardIterator std::search_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Integer __count,
    const _Tp & __val) [inline], [nodiscard], [constexpr]
```

Search a sequence for a number of consecutive values.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.

**Returns**

The first iterator `i` in the range `[__first, __last - __count)` such that `*(i+N) == __val` for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `count` consecutive elements equal to `__val`.

**search\_n()** [2/2]

```
template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
_FForwardIterator std::search_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Integer __count,
    const _Tp & __val,
    _BinaryPredicate __binary_pred) [inline], [nodiscard], [constexpr]
```

Search a sequence for a number of consecutive values using a predicate.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.
<code>__binary_pred</code>	A binary predicate.

**Returns**

The first iterator `i` in the range `[__first, __last - __count)` such that `__binary_pred(*(i+N), __val)` is true for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `__count` consecutive elements for which the predicate returns true.

### 3.1.5 Sorting

Collaboration diagram for Sorting:



#### Topics

- [Binary Search](#)
- [Heap](#)
- [Set Operations](#)

#### Functions

- `template<typename _Tp>`  
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)`
- `template<typename _BidirectionalIterator>`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↵  
last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↵  
last, _Compare __comp)`
- `template<typename _ForwardIterator>`  
`constexpr bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __↵  
__comp)`
- `template<typename _II1, typename _II2>`  
`constexpr bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`

- `template<typename _I1, typename _I2, typename _Compare>`  
`constexpr bool std::lexicographical\_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`
- `template<typename _InputIter1, typename _InputIter2, typename _Comp>`  
`constexpr auto std::lexicographical\_compare\_three\_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2, _InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))`
- `template<typename _Tp>`  
`constexpr const _Tp & std::max (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::max (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::min (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::min (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _ForwardIterator>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator>`  
`constexpr bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`



- `template<typename _RandomAccessIterator>`  
`constexpr void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _BidirectionalIterator>`  
`constexpr bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

### 3.1.5.1 Detailed Description

### 3.1.5.2 Function Documentation

#### `clamp()` [1/2]

```
template<typename _Tp>
constexpr const _Tp & std::clamp (
    const _Tp & __val,
    const _Tp & __lo,
    const _Tp & __hi) [nodiscard], [constexpr]
```

Returns the value clamped between lo and hi.

#### Parameters

<code>__val</code>	A value of arbitrary type.
<code>__lo</code>	A lower limit of arbitrary type.
<code>__hi</code>	An upper limit of arbitrary type.

#### Return values

<code>`__lo`</code>	if <code>__val &lt; __lo</code>
<code>`__hi`</code>	if <code>__hi &lt; __val</code>
<code>`__val`</code>	otherwise.

**Precondition**

`__Tp` is `LessThanComparable` and `(__hi < __lo)` is false.

References [max\(\)](#), and [min\(\)](#).

**clamp()** [2/2]

```
template<typename _Tp, typename _Compare>
const _Tp & std::clamp (
    const _Tp & __val,
    const _Tp & __lo,
    const _Tp & __hi,
    _Compare __comp) [nodiscard], [constexpr]
```

Returns the value clamped between lo and hi.

**Parameters**

<code>__val</code>	A value of arbitrary type.
<code>__lo</code>	A lower limit of arbitrary type.
<code>__hi</code>	An upper limit of arbitrary type.
<code>__comp</code>	A comparison functor.

**Return values**

<code>'__lo'</code>	if <code>__comp(__val, __lo)</code>
<code>'__hi'</code>	if <code>__comp(__hi, __val)</code>
<code>'__val'</code>	otherwise.

**Precondition**

`__comp(__hi, __lo)` is false.

References [max\(\)](#), and [min\(\)](#).

**inplace\_merge()** [1/2]

```
template<typename _BidirectionalIterator>
void std::inplace_merge (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last) [inline]
```

Merges two sorted ranges in place.

**Parameters**

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

**Returns**

Nothing.

Merges two sorted and consecutive ranges, `[__first,__middle)` and `[__middle,__last)`, and puts the result in `[__first,__last)`. The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes  $(\text{__last} - \text{__first}) - 1$  comparisons. Otherwise an  $N \log N$  algorithm is used, where  $N$  is `distance(__first,__last)`.

**inplace\_merge()** [2/2]

```
template<typename _BidirectionalIterator, typename _Compare>
void std::inplace_merge (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last,
    _Compare __comp) [inline]
```

Merges two sorted ranges in place.

**Parameters**

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A functor to use for comparisons.

**Returns**

Nothing.

Merges two sorted and consecutive ranges, `[__first,__middle)` and `[__middle,__last)`, and puts the result in `[__first,__last)`. The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes  $(\text{__last} - \text{__first}) - 1$  comparisons. Otherwise an  $N \log N$  algorithm is used, where  $N$  is `distance(__first,__last)`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

**is\_sorted()** [1/2]

```
template<typename _ForwardIterator>
bool std::is_sorted (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [nodiscard], [constexpr]
```

Determines whether the elements of a sequence are sorted.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

## Returns

True if the elements are sorted, false otherwise.

**is\_sorted()** [2/2]

```
template<typename _ForwardIterator, typename _Compare>
bool std::is_sorted (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Determines whether the elements of a sequence are sorted according to a comparison functor.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

## Returns

True if the elements are sorted, false otherwise.

**is\_sorted\_until()** [1/2]

```
template<typename _ForwardIterator>
_FowardIterator std::is_sorted_until (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [nodiscard], [constexpr]
```

Determines the end of a sorted sequence.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

## Returns

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

**is\_sorted\_until()** [2/2]

```
template<typename _ForwardIterator, typename _Compare>
_FForwardIterator std::is_sorted_until (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Determines the end of a sorted sequence using comparison functor.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

**Returns**

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

**lexicographical\_compare()** [1/2]

```
template<typename _II1, typename _II2>
bool std::lexicographical_compare (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2,
    _II2 __last2) [inline], [nodiscard], [constexpr]
```

Performs **dictionary** comparison on ranges.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

**Returns**

A boolean true or false.

*Returns true if the sequence of elements defined by the range `[first1,last1)` is lexicographically less than the sequence of elements defined by the range `[first2,last2)`. Returns false otherwise.* (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

**lexicographical\_compare()** [2/2]

```
template<typename _II1, typename _II2, typename _Compare>
bool std::lexicographical_compare (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2,
    _II2 __last2,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Performs **dictionary** comparison on ranges.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__comp</code>	A <a href="#">comparison functor</a> .

**Returns**

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

**lexicographical\_compare\_three\_way()**

```
template<typename _InputIter1, typename _InputIter2, typename _Comp>
auto std::lexicographical_compare_three_way (
    _InputIter1 __first1,
    _InputIter1 __last1,
    _InputIter2 __first2,
    _InputIter2 __last2,
    _Comp __comp) -> decltype(__comp(*__first1, *__first2)) [nodiscard], [constexpr]
```

Performs dictionary comparison on ranges.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__comp</code>	A <a href="#">comparison functor</a> .

**Returns**

The comparison category that `__comp(*__first1, *__first2)` returns.

Referenced by [operator<=>\(\)](#).

**max()** [1/2]

```
template<typename _Tp>
const _Tp & std::max (
    const _Tp & __a,
    const _Tp & __b) [inline], [nodiscard], [constexpr]
```

This does what you think it does.

**Parameters**

<a href="#"><code>__a</code></a>	A thing of arbitrary type.
<a href="#"><code>__b</code></a>	Another thing of arbitrary type.

**Returns**

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_nth\\_element\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partial\\_sum\\_linear\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partition\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_search\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_sequential\\_random\\_shuffle\(\)](#), [std::Deque\\_base<\\_Tp, \\_Alloc>::\\_M\\_initialize\\_buckets\(\)](#), [std::deque<\\_Tp, \\_Alloc>::\\_M\\_reallocate\\_map\(\)](#), [clamp\(\)](#), [clamp\(\)](#), [generate\\_canonical\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#), [std::shuffle\\_order\\_engine<\\_RandomNumberEngine, \\_\\_k>::operator\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#).

**max()** [2/2]

```
template<typename _Tp, typename _Compare>
const _Tp & std::max (
    const _Tp & __a,
    const _Tp & __b,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

This does what you think it does.

**Parameters**

<a href="#"><code>__a</code></a>	A thing of arbitrary type.
<a href="#"><code>__b</code></a>	Another thing of arbitrary type.
<a href="#"><code>__comp</code></a>	A <a href="#">comparison functor</a> .

**Returns**

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

**max\_element()** [1/2]

```
template<typename _ForwardIterator>
_FForwardIterator std::max_element (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [nodiscard], [constexpr]
```

Return the maximum element in a range.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

Iterator referencing the first instance of the largest value.

**max\_element()** [2/2]

```
template<typename _ForwardIterator, typename _Compare>
_FForwardIterator std::max_element (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Return the maximum element in a range using comparison functor.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

**Returns**

Iterator referencing the first instance of the largest value according to `__comp`.

**merge()** [1/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
_OutputIterator std::merge (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result) [inline], [constexpr]
```

Merges two sorted ranges.



## Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.

## Returns

An output iterator equal to `__result + (__last1 - __first1)`  
 • `(__last2 - __first2)`.

Merges the ranges `[__first1, __last1)` and `[__first2, __last2)` into the sorted range `[__result, __result + (__last1 - __first1) + (__last2 - __first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

**merge()** [2/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
_OutputIterator std::merge (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp) [inline], [constexpr]
```

Merges two sorted ranges.

## Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.
<code>__comp</code>	A functor to use for comparisons.

## Returns

An output iterator equal to `__result + (__last1 - __first1)`  
 • `(__last2 - __first2)`.

Merges the ranges `[__first1, __last1)` and `[__first2, __last2)` into the sorted range `[__result, __result + (__last1 - __first1) + (__last2 - __first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.



**Returns**

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

**min\_element() [1/2]**

```
template<typename _ForwardIterator>
_FowardIterator std::min_element (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [nodiscard], [constexpr]
```

Return the minimum element in a range.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

Iterator referencing the first instance of the smallest value.

**min\_element() [2/2]**

```
template<typename _ForwardIterator, typename _Compare>
_FowardIterator std::min_element (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Return the minimum element in a range using comparison functor.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

**Returns**

Iterator referencing the first instance of the smallest value according to `__comp`.

**minmax() [1/2]**

```
template<typename _Tp>
pair< const _Tp &, const _Tp & > std::minmax (
    const _Tp & __a,
    const _Tp & __b) [inline], [nodiscard], [constexpr]
```

Determines min and max at once as an ordered pair.

## Parameters

$\_a$	A thing of arbitrary type.
$\_b$	Another thing of arbitrary type.

## Returns

A pair( $\_b$ ,  $\_a$ ) if  $\_b$  is smaller than  $\_a$ , pair( $\_a$ ,  $\_b$ ) otherwise.

**minmax()** [2/2]

```
template<typename _Tp, typename _Compare>
pair< const _Tp &, const _Tp & > std::minmax (
    const _Tp & __a,
    const _Tp & __b,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Determines min and max at once as an ordered pair.

## Parameters

$\_a$	A thing of arbitrary type.
$\_b$	Another thing of arbitrary type.
$\_comp$	A <a href="#">comparison functor</a> .

## Returns

A pair( $\_b$ ,  $\_a$ ) if  $\_b$  is smaller than  $\_a$ , pair( $\_a$ ,  $\_b$ ) otherwise.

**minmax\_element()** [1/2]

```
template<typename _ForwardIterator>
pair< _ForwardIterator, _ForwardIterator > std::minmax_element (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [nodiscard], [constexpr]
```

Return a pair of iterators pointing to the minimum and maximum elements in a range.

## Parameters

$\_first$	Start of range.
$\_last$	End of range.

## Returns

make\_pair(m, M), where m is the first iterator i in [ $\_first$ ,  $\_last$ ) such that no other element in the range is smaller, and where M is the last iterator i in [ $\_first$ ,  $\_last$ ) such that no other element in the range is larger.

**minmax\_element()** [2/2]

```
template<typename _ForwardIterator, typename _Compare>
pair< _ForwardIterator, _ForwardIterator > std::minmax_element (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Return a pair of iterators pointing to the minimum and maximum elements in a range.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

**Returns**

make\_pair(m, M), where m is the first iterator i in [`__first`, `__last`) such that no other element in the range is smaller, and where M is the last iterator i in [`__first`, `__last`) such that no other element in the range is larger.

**next\_permutation()** [1/2]

```
template<typename _BidirectionalIterator>
bool std::next_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last) [inline], [constexpr]
```

Permute range into the next *dictionary* ordering.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

**next\_permutation()** [2/2]

```
template<typename _BidirectionalIterator, typename _Compare>
bool std::next_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _Compare __comp) [inline], [constexpr]
```

Permute range into the next *dictionary* ordering using comparison functor.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

## Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range `[__first, __last)` as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

**nth\_element()** [1/2]

```
template<typename _RandomAccessIterator>
void std::nth_element (
    _RandomAccessIterator __first,
    _RandomAccessIterator __nth,
    _RandomAccessIterator __last) [inline], [constexpr]
```

Sort a sequence just enough to find a particular position.

## Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.

## Returns

Nothing.

Rearranges the elements in the range `[__first, __last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator `i` in the range `[__first, __nth)` and any iterator `j` in the range `[__nth, __last)` it holds that `*j < *i` is false.

References [\\_\\_lg\(\)](#).

**nth\_element()** [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
void std::nth_element (
    _RandomAccessIterator __first,
    _RandomAccessIterator __nth,
    _RandomAccessIterator __last,
    _Compare __comp) [inline], [constexpr]
```

Sort a sequence just enough to find a particular position using a predicate for comparison.

**Parameters**

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

**Returns**

Nothing.

Rearranges the elements in the range `[__first, __last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator `i` in the range `[__first, __nth)` and any iterator `j` in the range `[__nth, __last)` it holds that `__comp(*j, *i)` is false.

References [\\_\\_lg\(\)](#).

**partial\_sort()** [1/2]

```
template<typename _RandomAccessIterator>
void std::partial_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __middle,
    _RandomAccessIterator __last) [inline], [constexpr]
```

Sort the smallest elements of a sequence.

**Parameters**

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

**Returns**

Nothing.

Sorts the smallest `(__middle - __first)` elements in the range `[first, last)` and moves them to the range `[__first, __middle)`. The order of the remaining elements in the range `[__middle, __last)` is unspecified. After the sort if `i` and `j` are iterators in the range `[__first, __middle)` such that `i` precedes `j` and `k` is an iterator in the range `[__middle, __last)` then `*j < *i` and `*k < *i` are both false.

**partial\_sort()** [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
void std::partial_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __middle,
    _RandomAccessIterator __last,
    _Compare __comp) [inline], [constexpr]
```

Sort the smallest elements of a sequence using a predicate for comparison.

## Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

## Returns

Nothing.

Sorts the smallest  $(\text{__middle} - \text{__first})$  elements in the range  $[\text{__first}, \text{__last})$  and moves them to the range  $[\text{__first}, \text{__middle})$ . The order of the remaining elements in the range  $[\text{__middle}, \text{__last})$  is unspecified. After the sort if  $i$  and  $j$  are iterators in the range  $[\text{__first}, \text{__middle})$  such that  $i$  precedes  $j$  and  $k$  is an iterator in the range  $[\text{__middle}, \text{__last})$  then  $*\text{__comp}(j, *i)$  and  $\text{__comp}(*k, *i)$  are both false.

**partial\_sort\_copy()** [1/2]

```
template<typename _InputIterator, typename _RandomAccessIterator>
_RandomAccessIterator std::partial_sort_copy (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __result_first,
    _RandomAccessIterator __result_last) [inline], [constexpr]
```

Copy the smallest elements of a sequence.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.

## Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest  $N$  values from the range  $[\text{__first}, \text{__last})$  to the range beginning at  $\text{__result\_first}$ , where the number of elements to be copied,  $N$ , is the smaller of  $(\text{__last} - \text{__first})$  and  $(\text{__result\_last} - \text{__result\_first})$ . After the sort if  $i$  and  $j$  are iterators in the range  $[\text{__result\_first}, \text{__result\_first} + N)$  such that  $i$  precedes  $j$  then  $*j < *i$  is false. The value returned is  $\text{__result\_first} + N$ .



**partial\_sort\_copy()** [2/2]

```
template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>
_RandomAccessIterator std::partial_sort_copy (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __result_first,
    _RandomAccessIterator __result_last,
    _Compare __comp) [inline], [constexpr]
```

Copy the smallest elements of a sequence using a predicate for comparison.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	Another input iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.
<code>__comp</code>	A comparison functor.

**Returns**

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest  $N$  values from the range  $[\text{__first}, \text{__last})$  to the range beginning at `result_`  
`__first`, where the number of elements to be copied,  $N$ , is the smaller of  $(\text{__last} - \text{__first})$  and  $(\text{__}  
`result_last - __result_first)`$ . After the sort if  $i$  and  $j$  are iterators in the range  $[\text{__result\_first},$   
`__result_first + N)` such that  $i$  precedes  $j$  then `__comp(*j, *i)` is false. The value returned is `__`  
`result_first + N`.

**prev\_permutation()** [1/2]

```
template<typename _BidirectionalIterator>
bool std::prev_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last) [inline], [constexpr]
```

Permute range into the previous *dictionary* ordering.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

**prev\_permutation()** [2/2]

```
template<typename _BidirectionalIterator, typename _Compare>
bool std::prev_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _Compare __comp) [inline], [constexpr]
```

Permute range into the previous *dictionary* ordering using comparison functor.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

**Returns**

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range `[__first, __last)` as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

**sort()** [1/2]

```
template<typename _RandomAccessIterator>
void std::sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last) [inline], [constexpr]
```

Sort the elements of a sequence.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

**Returns**

Nothing.

Sorts the elements in the range `[__first, __last)` in ascending order, such that for each iterator `i` in the range `[__first, __last - 1)`, `*(i+1) < *i` is false.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

**sort()** [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
void std::sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp) [inline], [constexpr]
```

Sort the elements of a sequence using a predicate for comparison.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

**Returns**

Nothing.

Sorts the elements in the range `[__first, __last)` in ascending order, such that `__comp(*(i+1), *i)` is false for every iterator `i` in the range `[__first, __last - 1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

**stable\_sort() [1/2]**

```
template<typename _RandomAccessIterator>
void std::stable_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last) [inline]
```

Sort the elements of a sequence, preserving the relative order of equivalent elements.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

**Returns**

Nothing.

Sorts the elements in the range `[__first, __last)` in ascending order, such that for each iterator `i` in the range `[__first, __last-1)`, `*(i+1) < *i` is false.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[__first, __last)` such that `x < y` is false and `y < x` is false will have the same relative ordering after calling `stable_sort()`.

**stable\_sort() [2/2]**

```
template<typename _RandomAccessIterator, typename _Compare>
void std::stable_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp) [inline]
```

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

## Returns

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator `i` in the range `[__first,__last-1)`, `__comp(*(i+1),*i)` is false.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `__comp(x,y)` is false and `__comp(y,x)` is false will have the same relative ordering after calling `stable_sort()`.

## 3.1.5.3 Binary Search

Collaboration diagram for Binary Search:



## Functions

- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr pair<_ForwardIterator, _ForwardIterator> std::equal\_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr pair<_ForwardIterator, _ForwardIterator> std::equal\_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

### 3.1.5.3.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by `equal_range` is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions the range.

### 3.1.5.3.2 Function Documentation

#### `binary_search()` [1/2]

```
template<typename _ForwardIterator, typename _Tp>
bool std::binary_search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val) [nodiscard], [constexpr]
```

Determines whether an element exists in a range.

#### Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

#### Returns

True if `__val` (or its equivalent) is in `[__first, __last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

**binary\_search()** [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare>
bool std::binary_search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp) [nodiscard], [constexpr]
```

Determines whether an element exists in a range.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

**Returns**

True if `__val` (or its equivalent) is in `[__first, __last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized `find` member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

**equal\_range()** [1/2]

```
template<typename _ForwardIterator, typename _Tp>
pair< _ForwardIterator, _ForwardIterator > std::equal_range (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val) [inline], [nodiscard], [constexpr]
```

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

**Returns**

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val),
               upper_bound(__first, __last, __val))
```

but does not actually call those functions.

**equal\_range()** [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare>
pair< _ForwardIterator, _ForwardIterator > std::equal_range (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

**Returns**

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val, __comp),
               upper_bound(__first, __last, __val, __comp))
```

but does not actually call those functions.

**lower\_bound()** [1/2]

```
template<typename _ForwardIterator, typename _Tp>
_FowardIterator std::lower_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val) [inline], [nodiscard], [constexpr]
```

Finds the first position in which `val` could be inserted without changing the ordering.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

**Returns**

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

**lower\_bound()** [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare>
_FowardIterator std::lower_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Finds the first position in which `__val` could be inserted without changing the ordering.

**Parameters**

<code>__first</code>	An iterator to the start of a sorted range.
<code>__last</code>	A past-the-end iterator for the sorted range.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

**Returns**

An iterator pointing to the first element *not less than* `__val`, or `end()` if every element is less than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

**upper\_bound()** [1/2]

```
template<typename _ForwardIterator, typename _Tp>
_FowardIterator std::upper_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val) [inline], [nodiscard], [constexpr]
```

Finds the last position in which `__val` could be inserted without changing the ordering.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

**Returns**

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

**upper\_bound()** [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare>
_FowardIterator std::upper_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Finds the last position in which `__val` could be inserted without changing the ordering.



## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

## Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

## 3.1.5.4 Heap

Collaboration diagram for Heap:



## Functions

- `template<typename _RandomAccessIterator>`  
`constexpr bool std::is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr bool std::is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator std::is\_heap\_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator std::is\_heap\_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::make\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::make\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↵`  
`comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::push\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::push\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↵`  
`comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↵`  
`comp)`

#### 3.1.5.4.1 Detailed Description

#### 3.1.5.4.2 Function Documentation

##### **is\_heap()** [1/2]

```
template<typename _RandomAccessIterator>
bool std::is_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last) [inline], [nodiscard], [constexpr]
```

Determines whether a range is a heap.

##### Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

##### Returns

True if range is a heap, false otherwise.

##### **is\_heap()** [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
bool std::is_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Determines whether a range is a heap using comparison functor.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor to use.

**Returns**

True if range is a heap, false otherwise.

References [distance\(\)](#).

**is\_heap\_until()** [1/2]

```
template<typename _RandomAccessIterator>
_RandomAccessIterator std::is_heap_until (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last) [inline], [nodiscard], [constexpr]
```

Search the end of a heap.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [`__first`, `__last`) for which the range [`__first`, *i*) is a heap.

References [distance\(\)](#).

**is\_heap\_until()** [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
_RandomAccessIterator std::is_heap_until (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Search the end of a heap using comparison functor.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor to use.

## Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is a heap. Comparisons are made using `__comp`.

References [distance\(\)](#).

**make\_heap()** [1/2]

```
template<typename _RandomAccessIterator>
void std::make_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last) [inline], [constexpr]
```

Construct a heap over a range.

## Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

This operation makes the elements in `[__first, __last)` into a heap.

**make\_heap()** [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
void std::make_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp) [inline], [constexpr]
```

Construct a heap over a range using comparison functor.

## Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation makes the elements in `[__first, __last)` into a heap. Comparisons are made using `__comp`.

**pop\_heap()** [1/2]

```
template<typename _RandomAccessIterator>
void std::pop_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last) [inline], [constexpr]
```

Pop an element off a heap.

**Parameters**

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

**Precondition**

[`__first`, `__last`) is a valid, non-empty range.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and [`__first`,`__last-1`) is made into a heap.

**pop\_heap()** [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
void std::pop_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp) [inline], [constexpr]
```

Pop an element off a heap using comparison functor.

**Parameters**

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and [`__first`,`__last-1`) is made into a heap. Comparisons are made using `comp`.

**push\_heap()** [1/2]

```
template<typename _RandomAccessIterator>
void std::push_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last) [inline], [constexpr]
```

Push an element onto a heap.

## Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.

This operation pushes the element at `last-1` onto the valid heap over the range `[__first,__last-1)`. After completion, `[__first,__last)` is a valid heap.

**push\_heap()** [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
void std::push_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp) [inline], [constexpr]
```

Push an element onto a heap using comparison functor.

## Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.
<code>__comp</code>	Comparison functor.

This operation pushes the element at `__last-1` onto the valid heap over the range `[__first,__last-1)`. After completion, `[__first,__last)` is a valid heap. Compare operations are performed using `comp`.

**sort\_heap()** [1/2]

```
template<typename _RandomAccessIterator>
void std::sort_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last) [inline], [constexpr]
```

Sort a heap.

## Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

This operation sorts the valid heap in the range `[__first,__last)`.

**sort\_heap()** [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
void std::sort_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp) [inline], [constexpr]
```

Sort a heap using comparison functor.

## Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation sorts the valid heap in the range `[__first,__last)`. Comparisons are made using `__comp`.

## 3.1.5.5 Set Operations

Collaboration diagram for Set Operations:



## Functions

- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare>`  
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

#### 3.1.5.5.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

#### 3.1.5.5.2 Function Documentation

##### **includes()** [1/2]

```
template<typename _InputIterator1, typename _InputIterator2>
bool std::includes (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2) [inline], [nodiscard], [constexpr]
```

Determines whether all elements of a sequence exists in a range.

##### Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.

##### Returns

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

##### **includes()** [2/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
bool std::includes (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Determines whether all elements of a sequence exists in a range using comparison.



**Parameters**

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.
<code>__comp</code>	Comparison function to use.

**Returns**

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)` according to `comp`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`, using `comp` to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

**set\_difference()** [1/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
_OutputIterator std::set_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result) [inline], [constexpr]
```

Return the difference of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

**set\_difference()** [2/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
_OutputIterator std::set_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp) [inline], [constexpr]
```

Return the difference of two sorted ranges using comparison functor.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to `__comp`, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

**set\_intersection()** [1/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
_OutputIterator std::set_intersection (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result) [inline], [constexpr]
```

Return the intersection of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.

<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

#### **set\_intersection()** [2/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
_OutputIterator std::set_intersection (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp) [inline], [constexpr]
```

Return the intersection of two sorted ranges using comparison functor.

#### Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that iterator advances. If an element is contained in both ranges according to `__comp`, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

**set\_symmetric\_difference()** [1/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
_OutputIterator std::set_symmetric_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result) [inline], [constexpr]
```

Return the symmetric difference of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

**set\_symmetric\_difference()** [2/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
_OutputIterator std::set_symmetric_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp) [inline], [constexpr]
```

Return the symmetric difference of two sorted ranges using comparison functor.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `comp`, that element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

**set\_union()** [1/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
_OutputIterator std::set_union (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result) [inline], [constexpr]
```

Return the union of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

**set\_union()** [2/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename <←
_Compare>
_OutputIterator std::set_union (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp) [inline], [constexpr]
```

Return the union of two sorted ranges using a comparison functor.

## Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

## Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that element is copied and the iterator advanced. If an equivalent element according to `__comp` is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

## 3.2 Atomics

## Classes

- struct [std::atomic<\\_Tp>](#)
- struct [std::atomic<\\_Tp\\*>](#)
- struct [std::atomic\\_flag](#)

## Macros

- `#define` [ATOMIC\\_BOOL\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_CHAR16\\_T\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_CHAR32\\_T\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_CHAR\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_FLAG\\_INIT](#)
- `#define` [ATOMIC\\_INT\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_LLONG\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_LONG\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_POINTER\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_SHORT\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_VAR\\_INIT\(\\_VI\)](#)
- `#define` [ATOMIC\\_WCHAR\\_T\\_LOCK\\_FREE](#)

## Typedefs

- typedef unsigned char **std::\_\_atomic\_flag\_data\_type**
- typedef [atomic](#)< bool > [std::atomic\\_bool](#)
- typedef [atomic](#)< char > [std::atomic\\_char](#)
- typedef [atomic](#)< char16\_t > [std::atomic\\_char16\\_t](#)
- typedef [atomic](#)< char32\_t > [std::atomic\\_char32\\_t](#)
- typedef [atomic](#)< int > [std::atomic\\_int](#)
- typedef [atomic](#)< int16\_t > [std::atomic\\_int16\\_t](#)
- typedef [atomic](#)< int32\_t > [std::atomic\\_int32\\_t](#)
- typedef [atomic](#)< int64\_t > [std::atomic\\_int64\\_t](#)
- typedef [atomic](#)< int8\_t > [std::atomic\\_int8\\_t](#)
- typedef [atomic](#)< int\_fast16\_t > [std::atomic\\_int\\_fast16\\_t](#)
- typedef [atomic](#)< int\_fast32\_t > [std::atomic\\_int\\_fast32\\_t](#)
- typedef [atomic](#)< int\_fast64\_t > [std::atomic\\_int\\_fast64\\_t](#)
- typedef [atomic](#)< int\_fast8\_t > [std::atomic\\_int\\_fast8\\_t](#)
- typedef [atomic](#)< int\_least16\_t > [std::atomic\\_int\\_least16\\_t](#)
- typedef [atomic](#)< int\_least32\_t > [std::atomic\\_int\\_least32\\_t](#)
- typedef [atomic](#)< int\_least64\_t > [std::atomic\\_int\\_least64\\_t](#)
- typedef [atomic](#)< int\_least8\_t > [std::atomic\\_int\\_least8\\_t](#)
- typedef [atomic](#)< intmax\_t > [std::atomic\\_intmax\\_t](#)
- typedef [atomic](#)< intptr\_t > [std::atomic\\_intptr\\_t](#)
- typedef [atomic](#)< long long > [std::atomic\\_llong](#)
- typedef [atomic](#)< long > [std::atomic\\_long](#)
- typedef [atomic](#)< ptrdiff\_t > [std::atomic\\_ptrdiff\\_t](#)
- typedef [atomic](#)< signed char > [std::atomic\\_schar](#)
- typedef [atomic](#)< short > [std::atomic\\_short](#)
- typedef [atomic](#)< size\_t > [std::atomic\\_size\\_t](#)
- typedef [atomic](#)< unsigned char > [std::atomic\\_uchar](#)
- typedef [atomic](#)< unsigned int > [std::atomic\\_uint](#)
- typedef [atomic](#)< uint16\_t > [std::atomic\\_uint16\\_t](#)
- typedef [atomic](#)< uint32\_t > [std::atomic\\_uint32\\_t](#)
- typedef [atomic](#)< uint64\_t > [std::atomic\\_uint64\\_t](#)
- typedef [atomic](#)< uint8\_t > [std::atomic\\_uint8\\_t](#)
- typedef [atomic](#)< uint\_fast16\_t > [std::atomic\\_uint\\_fast16\\_t](#)
- typedef [atomic](#)< uint\_fast32\_t > [std::atomic\\_uint\\_fast32\\_t](#)
- typedef [atomic](#)< uint\_fast64\_t > [std::atomic\\_uint\\_fast64\\_t](#)
- typedef [atomic](#)< uint\_fast8\_t > [std::atomic\\_uint\\_fast8\\_t](#)
- typedef [atomic](#)< uint\_least16\_t > [std::atomic\\_uint\\_least16\\_t](#)
- typedef [atomic](#)< uint\_least32\_t > [std::atomic\\_uint\\_least32\\_t](#)
- typedef [atomic](#)< uint\_least64\_t > [std::atomic\\_uint\\_least64\\_t](#)
- typedef [atomic](#)< uint\_least8\_t > [std::atomic\\_uint\\_least8\\_t](#)
- typedef [atomic](#)< uintmax\_t > [std::atomic\\_uintmax\\_t](#)
- typedef [atomic](#)< uintptr\_t > [std::atomic\\_uintptr\\_t](#)
- typedef [atomic](#)< unsigned long long > [std::atomic\\_ullong](#)
- typedef [atomic](#)< unsigned long > [std::atomic\\_ulong](#)
- typedef [atomic](#)< unsigned short > [std::atomic\\_ushort](#)
- typedef [atomic](#)< wchar\_t > [std::atomic\\_wchar\\_t](#)

## Enumerations

- enum class `std::memory_order` : int {  
`relaxed` , `consume` , `acquire` , `release` ,  
`acq_rel` , `seq_cst` }

## Functions

- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_strong** (atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \*\_\_i1, \_\_atomic\_val\_t< \_ITp > \_\_i2) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_strong** (volatile atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \*\_\_i1, \_\_atomic\_val\_t< \_ITp > \_\_i2) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_strong\_explicit** (atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \*\_\_i1, \_\_atomic\_val\_t< \_ITp > \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_strong\_explicit** (volatile atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \*\_\_i1, \_\_atomic\_val\_t< \_ITp > \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_weak** (atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \*\_\_i1, \_\_atomic\_val\_t< \_ITp > \_\_i2) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_weak** (volatile atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \*\_\_i1, \_\_atomic\_val\_t< \_ITp > \_\_i2) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_weak\_explicit** (atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \*\_\_i1, \_\_atomic\_val\_t< \_ITp > \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_weak\_explicit** (volatile atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \*\_\_i1, \_\_atomic\_val\_t< \_ITp > \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2) noexcept
- template<typename \_ITp>  
\_\_ITp **std::atomic\_exchange** (atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \_\_i) noexcept
- template<typename \_ITp>  
\_\_ITp **std::atomic\_exchange** (volatile atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \_\_i) noexcept
- template<typename \_ITp>  
\_\_ITp **std::atomic\_exchange\_explicit** (atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \_\_i, memory\_order \_\_m) noexcept
- template<typename \_ITp>  
\_\_ITp **std::atomic\_exchange\_explicit** (volatile atomic< \_ITp > \*\_\_a, \_\_atomic\_val\_t< \_ITp > \_\_i, memory\_order \_\_m) noexcept
- template<typename \_ITp>  
\_\_ITp **std::atomic\_fetch\_add** (atomic< \_ITp > \*\_\_a, \_\_atomic\_diff\_t< \_ITp > \_\_i) noexcept
- template<typename \_ITp>  
\_\_ITp **std::atomic\_fetch\_add** (volatile atomic< \_ITp > \*\_\_a, \_\_atomic\_diff\_t< \_ITp > \_\_i) noexcept
- template<typename \_ITp>  
\_\_ITp **std::atomic\_fetch\_add\_explicit** (atomic< \_ITp > \*\_\_a, \_\_atomic\_diff\_t< \_ITp > \_\_i, memory\_order \_\_m) noexcept
- template<typename \_ITp>  
\_\_ITp **std::atomic\_fetch\_add\_explicit** (volatile atomic< \_ITp > \*\_\_a, \_\_atomic\_diff\_t< \_ITp > \_\_i, memory\_order \_\_m) noexcept



- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void std::atomic_flag_clear (atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void std::atomic_init (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void std::atomic_init (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`

- `template<typename _ITp>`  
`bool std::atomic_is_lock_free (const atomic<_ITp> *__a) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_is_lock_free (const volatile atomic<_ITp> *__a) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load (const atomic<_ITp> *__a) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load (const volatile atomic<_ITp> *__a) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load_explicit (const atomic<_ITp> *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load_explicit (const volatile atomic<_ITp> *__a, memory_order __m) noexcept`
- `void std::atomic_signal_fence (memory_order __m) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store_explicit (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store_explicit (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `void std::atomic_thread_fence (memory_order __m) noexcept`
- `template<typename _Tp>`  
`_Tp std::kill_dependency (_Tp __y) noexcept`
- `constexpr memory_order std::operator& (memory_order __m, __memory_order_modifier __mod) noexcept`
- `constexpr memory_order std::operator| (memory_order __m, __memory_order_modifier __mod) noexcept`

## Variables

- `constexpr memory_order std::memory_order_acq_rel`
- `constexpr memory_order std::memory_order_acquire`
- `constexpr memory_order std::memory_order_consume`
- `constexpr memory_order std::memory_order_relaxed`
- `constexpr memory_order std::memory_order_release`
- `constexpr memory_order std::memory_order_seq_cst`

### 3.2.1 Detailed Description

Components for performing atomic operations.

### 3.2.2 Macro Definition Documentation

#### ATOMIC\_BOOL\_LOCK\_FREE

```
#define ATOMIC_BOOL_LOCK_FREE
```

Lock-free property.

0 indicates that the types are never lock-free. 1 indicates that the types are sometimes lock-free. 2 indicates that the types are always lock-free.

### 3.2.3 Typedef Documentation

#### **atomic\_bool**

```
typedef atomic<bool> std::atomic_bool
```

`atomic_bool`

#### **atomic\_char**

```
typedef atomic<char> std::atomic_char
```

`atomic_char`

#### **atomic\_char16\_t**

```
typedef atomic<char16_t> std::atomic_char16_t
```

`atomic_char16_t`

#### **atomic\_char32\_t**

```
typedef atomic<char32_t> std::atomic_char32_t
```

`atomic_char32_t`

#### **atomic\_int**

```
typedef atomic<int> std::atomic_int
```

`atomic_int`

#### **atomic\_int16\_t**

```
typedef atomic<int16_t> std::atomic_int16_t
```

`atomic_int16_t`

#### **atomic\_int32\_t**

```
typedef atomic<int32_t> std::atomic_int32_t
```

`atomic_int32_t`

**atomic\_int64\_t**

```
typedef atomic<int64_t> std::atomic_int64_t
```

atomic\_int64\_t

**atomic\_int8\_t**

```
typedef atomic<int8_t> std::atomic_int8_t
```

atomic\_int8\_t

**atomic\_int\_fast16\_t**

```
typedef atomic<int_fast16_t> std::atomic_int_fast16_t
```

atomic\_int\_fast16\_t

**atomic\_int\_fast32\_t**

```
typedef atomic<int_fast32_t> std::atomic_int_fast32_t
```

atomic\_int\_fast32\_t

**atomic\_int\_fast64\_t**

```
typedef atomic<int_fast64_t> std::atomic_int_fast64_t
```

atomic\_int\_fast64\_t

**atomic\_int\_fast8\_t**

```
typedef atomic<int_fast8_t> std::atomic_int_fast8_t
```

atomic\_int\_fast8\_t

**atomic\_int\_least16\_t**

```
typedef atomic<int_least16_t> std::atomic_int_least16_t
```

atomic\_int\_least16\_t

**atomic\_int\_least32\_t**

```
typedef atomic<int_least32_t> std::atomic\_int\_least32\_t
```

[atomic\\_int\\_least32\\_t](#)

**atomic\_int\_least64\_t**

```
typedef atomic<int_least64_t> std::atomic\_int\_least64\_t
```

[atomic\\_int\\_least64\\_t](#)

**atomic\_int\_least8\_t**

```
typedef atomic<int_least8_t> std::atomic\_int\_least8\_t
```

[atomic\\_int\\_least8\\_t](#)

**atomic\_intmax\_t**

```
typedef atomic<intmax_t> std::atomic\_intmax\_t
```

[atomic\\_intmax\\_t](#)

**atomic\_intptr\_t**

```
typedef atomic<intptr_t> std::atomic\_intptr\_t
```

[atomic\\_intptr\\_t](#)

**atomic\_llong**

```
typedef atomic<long long> std::atomic\_llong
```

[atomic\\_llong](#)

**atomic\_long**

```
typedef atomic<long> std::atomic\_long
```

[atomic\\_long](#)

**atomic\_ptrdiff\_t**

```
typedef atomic<ptrdiff_t> std::atomic_ptrdiff_t
```

atomic\_ptrdiff\_t

**atomic\_schar**

```
typedef atomic<signed char> std::atomic_schar
```

atomic\_schar

**atomic\_short**

```
typedef atomic<short> std::atomic_short
```

atomic\_short

**atomic\_size\_t**

```
typedef atomic<size_t> std::atomic_size_t
```

atomic\_size\_t

**atomic\_uchar**

```
typedef atomic<unsigned char> std::atomic_uchar
```

atomic\_uchar

**atomic\_uint**

```
typedef atomic<unsigned int> std::atomic_uint
```

atomic\_uint

**atomic\_uint16\_t**

```
typedef atomic<uint16_t> std::atomic_uint16_t
```

atomic\_uint16\_t

**atomic\_uint32\_t**

```
typedef atomic<uint32_t> std::atomic_uint32_t
```

atomic\_uint32\_t

**atomic\_uint64\_t**

```
typedef atomic<uint64_t> std::atomic_uint64_t
```

atomic\_uint64\_t

**atomic\_uint8\_t**

```
typedef atomic<uint8_t> std::atomic_uint8_t
```

atomic\_uint8\_t

**atomic\_uint\_fast16\_t**

```
typedef atomic<uint_fast16_t> std::atomic_uint_fast16_t
```

atomic\_uint\_fast16\_t

**atomic\_uint\_fast32\_t**

```
typedef atomic<uint_fast32_t> std::atomic_uint_fast32_t
```

atomic\_uint\_fast32\_t

**atomic\_uint\_fast64\_t**

```
typedef atomic<uint_fast64_t> std::atomic_uint_fast64_t
```

atomic\_uint\_fast64\_t

**atomic\_uint\_fast8\_t**

```
typedef atomic<uint_fast8_t> std::atomic_uint_fast8_t
```

atomic\_uint\_fast8\_t

**atomic\_uint\_least16\_t**

```
typedef atomic<uint_least16_t> std::atomic_uint_least16_t
```

`atomic_uint_least16_t`

**atomic\_uint\_least32\_t**

```
typedef atomic<uint_least32_t> std::atomic_uint_least32_t
```

`atomic_uint_least32_t`

**atomic\_uint\_least64\_t**

```
typedef atomic<uint_least64_t> std::atomic_uint_least64_t
```

`atomic_uint_least64_t`

**atomic\_uint\_least8\_t**

```
typedef atomic<uint_least8_t> std::atomic_uint_least8_t
```

`atomic_uint_least8_t`

**atomic\_uintmax\_t**

```
typedef atomic<uintmax_t> std::atomic_uintmax_t
```

`atomic_uintmax_t`

**atomic\_uintptr\_t**

```
typedef atomic<uintptr_t> std::atomic_uintptr_t
```

`atomic_uintptr_t`

**atomic\_ullong**

```
typedef atomic<unsigned long long> std::atomic_ullong
```

`atomic_ullong`



**atomic\_ulong**

```
typedef atomic<unsigned long> std::atomic_ulong
```

`atomic_ulong`

**atomic\_ushort**

```
typedef atomic<unsigned short> std::atomic_ushort
```

`atomic_ushort`

**atomic\_wchar\_t**

```
typedef atomic<wchar_t> std::atomic_wchar_t
```

`atomic_wchar_t`

**3.2.4 Enumeration Type Documentation****memory\_order**

```
enum class std::memory_order : int [strong]
```

Enumeration for `memory_order`.

**3.2.5 Function Documentation****kill\_dependency()**

```
template<typename _Tp>  
_Tp std::kill_dependency (  
    _Tp __y) [inline], [noexcept]
```

`kill_dependency`

### 3.3 Concurrency

Collaboration diagram for Concurrency:



#### Topics

- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

#### 3.3.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

#### 3.3.2 Condition Variables

Collaboration diagram for Condition Variables:



## Classes

- class `std::condition_variable`
- class `std::condition_variable_any`

## Enumerations

- enum class `std::cv_status` { `no_timeout` , `timeout` }

## Functions

- void `std::notify_all_at_thread_exit` (`condition_variable` &, `unique_lock`< `mutex` >)

### 3.3.2.1 Detailed Description

Classes for `condition_variable` support.

### 3.3.2.2 Enumeration Type Documentation

#### `cv_status`

```
enum class std::cv_status [strong]
```

`cv_status`

### 3.3.3 Futures

Collaboration diagram for Futures:



## Classes

- class `std::__basic_future< _Res >`
- class `std::future< _Res >`
- class `std::future< _Res & >`
- class `std::future< void >`
- class `std::future_error`
- struct `std::is_error_code_enum< future_errc >`
- class `std::packaged_task< _Res(_ArgTypes...)>`
- class `std::promise< _Res >`
- class `std::promise< _Res & >`
- class `std::promise< void >`
- class `std::shared_future< _Res >`
- class `std::shared_future< _Res & >`
- class `std::shared_future< void >`

## Enumerations

- enum class `std::future_errc` { `future_already_retrieved` , `promise_already_satisfied` , `no_state` , `broken_←  
promise` }
- enum class `std::future_status` { `ready` , `timeout` , `deferred` }
- enum class `std::launch` { `async` , `deferred` }

## Functions

- `std::__basic_future< _Res >::__basic_future` (const `shared_future< _Res >` &) noexcept
- `std::__basic_future< _Res >::__basic_future` (`future< _Res >` &&) noexcept
- `std::__basic_future< _Res >::__basic_future` (`shared_future< _Res >` &&) noexcept
- template<typename \_Fn, typename... \_Args>  
`future< __async_result_of< _Fn, _Args... > > std::async` (\_Fn &&\_\_fn, \_Args &&... \_\_args)
- template<typename \_Fn, typename... \_Args>  
`future< __async_result_of< _Fn, _Args... > > std::async` (`launch` \_\_policy, \_Fn &&\_\_fn, \_Args &&... \_\_args)
- const `error_category` & `std::future_category` () noexcept
- `error_code` `std::make_error_code` (`future_errc` \_\_errc) noexcept
- `error_condition` `std::make_error_condition` (`future_errc` \_\_errc) noexcept
- constexpr `launch` `std::operator&` (`launch` \_\_x, `launch` \_\_y) noexcept
- constexpr `launch` & `std::operator&=` (`launch` &\_\_x, `launch` \_\_y) noexcept
- constexpr `launch` `std::operator^` (`launch` \_\_x, `launch` \_\_y) noexcept
- constexpr `launch` & `std::operator^=` (`launch` &\_\_x, `launch` \_\_y) noexcept
- constexpr `launch` `std::operator|` (`launch` \_\_x, `launch` \_\_y) noexcept
- constexpr `launch` & `std::operator|=` (`launch` &\_\_x, `launch` \_\_y) noexcept
- constexpr `launch` `std::operator~` (`launch` \_\_x) noexcept
- template<typename \_Fun, typename \_Signature = \_\_function\_guide\_t<\_Fun, decltype(&\_Fun::operator())>>  
`std::packaged_task` (\_Fun) -> `packaged_task< _Signature >`
- template<typename \_Res, typename... \_ArgTypes>  
`std::packaged_task` (\_Res\*)(\_ArgTypes...) -> `packaged_task< _Res(_ArgTypes...)>`
- `shared_future< _Res >` `std::future< _Res >::share` () noexcept
- `shared_future< _Res & >` `std::future< _Res & >::share` () noexcept
- `shared_future< void >` `std::future< void >::share` () noexcept
- template<typename \_Res, typename... \_ArgTypes>  
void `std::swap` (`packaged_task< _Res(_ArgTypes...)>` &\_\_x, `packaged_task< _Res(_ArgTypes...)>` &\_\_y) noexcept
- template<typename \_Res>  
void `std::swap` (`promise< _Res >` &\_\_x, `promise< _Res >` &\_\_y) noexcept

### 3.3.3.1 Detailed Description

Futures and promises provide support for retrieving the result from an asynchronous function, e.g. one that is running in another thread. A `std::future` represents an asynchronous result that will become ready at some later time. A consumer can wait on a future until the result is ready to be accessed.

Since

C++11

### 3.3.3.2 Enumeration Type Documentation

#### **future\_errc**

```
enum class std::future_errc [strong]
```

Error code for futures.

#### **future\_status**

```
enum class std::future_status [strong]
```

Status code for futures.

#### **launch**

```
enum class std::launch [strong]
```

Launch code for futures.

### 3.3.3.3 Function Documentation

#### **async()** [1/2]

```
template<typename _Fn, typename... _Args>
future< __async_result_of< _Fn, _Args... > > std::async (
    _Fn && __fn,
    _Args &&... __args) [inline], [nodiscard]
```

async, potential overload

**async()** [2/2]

```
template<typename _Fn, typename... _Args>
future< __async_result_of< _Fn, _Args... > > std::async (
    launch __policy,
    _Fn && __fn,
    _Args &&... __args) [nodiscard]
```

async

**future\_category()**

```
const error_category & std::future_category () [noexcept]
```

Points to a statically-allocated object derived from error\_category.

**make\_error\_code()**

```
error_code std::make_error_code (
    future_errc __errc) [inline], [noexcept]
```

Overload of make\_error\_code for future\_errc.

Referenced by [std::filesystem::filesystem\\_error::what\(\)](#).

**make\_error\_condition()**

```
error_condition std::make_error_condition (
    future_errc __errc) [inline], [noexcept]
```

Overload of make\_error\_condition for future\_errc.

**swap()**

```
template<typename _Res, typename... _ArgTypes>
void std::swap (
    packaged_task< _Res(_ArgTypes...)> & __x,
    packaged_task< _Res(_ArgTypes...)> & __y) [inline], [noexcept]
```

swap

### 3.3.4 Mutexes

Collaboration diagram for Mutexes:



#### Classes

- struct `std::adopt_lock_t`
- struct `std::defer_lock_t`
- class `std::lock_guard<_Mutex>`
- class `std::mutex`
- struct `std::once_flag`
- class `std::recursive_mutex`
- class `std::recursive_timed_mutex`
- class `std::shared_lock<_Mutex>`
- class `std::shared_timed_mutex`
- class `std::timed_mutex`
- struct `std::try_to_lock_t`
- class `std::unique_lock<_Mutex>`

#### Functions

- void `std::__once_proxy` (void)
- template<typename \_Callable, typename... \_Args>  
void `std::call_once` (`once_flag` &\_\_once, \_Callable &&\_\_f, \_Args &&... \_\_args)
- template<typename \_L1, typename \_L2, typename... \_L3>  
void `std::lock` (\_L1 &\_\_l1, \_L2 &\_\_l2, \_L3 &... \_\_l3)
- template<typename \_L1, typename \_L2, typename... \_L3>  
int `std::try_lock` (\_L1 &\_\_l1, \_L2 &\_\_l2, \_L3 &... \_\_l3)

#### Variables

- constexpr `adopt_lock_t` `std::adopt_lock`
- constexpr `defer_lock_t` `std::defer_lock`
- constexpr `try_to_lock_t` `std::try_to_lock`

#### 3.3.4.1 Detailed Description

Classes for mutex support.

### 3.3.4.2 Function Documentation

#### call\_once()

```
template<typename _Callable, typename... _Args>
void std::call_once (
    once_flag & __once,
    _Callable && __f,
    _Args &&... __args)
```

Invoke a callable and synchronize with other calls using the same flag.

#### lock()

```
template<typename _L1, typename _L2, typename... _L3>
void std::lock (
    _L1 & __l1,
    _L2 & __l2,
    _L3 &... __l3)
```

Generic lock.

##### Parameters

$\leftrightarrow$ _l1	Meets Lockable requirements (try_lock() may throw).
$\leftrightarrow$ _l2	Meets Lockable requirements (try_lock() may throw).
$\leftrightarrow$ _l3	Meets Lockable requirements (try_lock() may throw).

##### Exceptions

<i>An</i>	exception thrown by an argument's lock() or try_lock() member.
-----------	--

##### Postcondition

All arguments are locked.

All arguments are locked via a sequence of calls to lock(), try\_lock() and unlock(). If this function exits via an exception any locks that were obtained will be released.

#### try\_lock()

```
template<typename _L1, typename _L2, typename... _L3>
int std::try_lock (
    _L1 & __l1,
    _L2 & __l2,
    _L3 &... __l3) [inline], [nodiscard]
```

Generic try\_lock.



**Parameters**

$\leftrightarrow$ _l1	Meets Lockable requirements (try_lock() may throw).
$\leftrightarrow$ _l2	Meets Lockable requirements (try_lock() may throw).
$\leftrightarrow$ _l3	Meets Lockable requirements (try_lock() may throw).

**Returns**

Returns -1 if all try\_lock() calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

**Postcondition**

Either all arguments are locked, or none will be.

Sequentially calls try\_lock() on each argument.

**3.3.4.3 Variable Documentation****adopt\_lock**

```
adopt_lock_t std::adopt_lock [inline], [constexpr]
```

Tag used to make a scoped lock take ownership of a locked mutex.

**defer\_lock**

```
defer_lock_t std::defer_lock [inline], [constexpr]
```

Tag used to prevent a scoped lock from acquiring ownership of a mutex.

**try\_to\_lock**

```
try_to_lock_t std::try_to_lock [inline], [constexpr]
```

Tag used to prevent a scoped lock from blocking if a mutex is locked.

### 3.3.5 Threads

Collaboration diagram for Threads:



#### Namespaces

- namespace `std::this_thread`

#### Classes

- struct `std::hash< thread::id >`
- class `std::thread`

#### Functions

- bool **operator==** (`thread::id __x`, `thread::id __y`) noexcept
- void **swap** (`thread &__x`, `thread &__y`) noexcept
- strong\_ordering **operator<=>** (`thread::id __x`, `thread::id __y`) noexcept
- template<class `_CharT`, class `_Traits`>  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)`

#### 3.3.5.1 Detailed Description

Since

C++11

Classes for thread support.

## 3.4 Containers

Collaboration diagram for Containers:



### Topics

- [Associative](#)
- [Sequences](#)
- [Unordered Associative](#)

### 3.4.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in `tables`.

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

### 3.4.2 Associative

Collaboration diagram for Associative:



#### Classes

- class `std::map<_Key, _Tp, _Compare, _Alloc>`
- class `std::multimap<_Key, _Tp, _Compare, _Alloc>`
- class `std::multiset<_Key, _Compare, _Alloc>`
- class `std::set<_Key, _Compare, _Alloc>`

#### 3.4.2.1 Detailed Description

Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

### 3.4.3 Sequences

Collaboration diagram for Sequences:



## Files

- file `inplace_vector`

## Classes

- struct `std::array<_Tp, _Nm>`
- class `std::basic_string<_CharT, _Traits, _Alloc>`
- class `std::basic_string_view<_CharT, _Traits>`
- class `std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits>`
- class `std::deque<_Tp, _Alloc>`
- class `std::forward_list<_Tp, _Alloc>`
- class `std::list<_Tp, _Alloc>`
- class `std::priority_queue<_Tp, _Sequence, _Compare>`
- class `std::queue<_Tp, _Sequence>`
- class `std::stack<_Tp, _Sequence>`
- class `std::vector<_Tp, _Alloc>`
- class `std::vector<bool, _Alloc>`

### 3.4.3.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in `tables`.

### 3.4.4 Unordered Associative

Collaboration diagram for Unordered Associative:



## Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`

### 3.4.4.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).

## 3.5 Diagnostics

Collaboration diagram for Diagnostics:



## Topics

- [Exceptions](#)

## Classes

- class `std::error_category`
- class `std::error_code`
- class `std::error_condition`
- struct `std::is_error_code_enum<_Tp>`
- struct `std::is_error_condition_enum<_Tp>`

## Functions

- `const error_category & std::generic_category ()` noexcept
- `error_code make_error_code (errc __e)` noexcept
- `error_condition make_error_condition (errc __e)` noexcept
- `template<typename _CharT, typename _Traits>  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code  
&__e)`
- `strong_ordering operator<=> (const error_code &__lhs, const error_code &__rhs)` noexcept
- `strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs)` noexcept
- `bool operator== (const error_code &__lhs, const error_code &__rhs)` noexcept
- `bool operator== (const error_code &__lhs, const error_condition &__rhs)` noexcept
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs)` noexcept
- `const error_category & std::system_category ()` noexcept

## Variables

- `template<typename _Tp>  
constexpr bool std::is_error_code_enum_v`
- `template<typename _Tp>  
constexpr bool std::is_error_condition_enum_v`

### 3.5.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

### 3.5.2 Function Documentation

#### **generic\_category()**

```
const error_category & std::generic_category () [noexcept]
```

Error category for `errno` error codes.

#### **make\_error\_code()**

```
error_code make_error_code (  
    errc __e) [related]
```

Create an `error_code` representing a standard `errc` condition.

The `std::errc` constants correspond to `errno` macros and so use the generic category.

Since

C++11

**make\_error\_condition()**

```
error_condition make_error_condition (
    errc __e) [related]
```

Create an `error_condition` representing a standard `errc` condition.

The `std::errc` constants correspond to `errno` macros and so use the generic category.

Since

C++11

**operator<<()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const error_code & __e) [related]
```

Write a `std::error_code` to an ostream.

Since

C++11

**operator<=>() [1/2]**

```
strong_ordering operator<=> (
    const error_code & __lhs,
    const error_code & __rhs) [related]
```

Ordered comparison for `std::error_code`.

This defines a total order by comparing the categories, and then if they are equal comparing the values.

Since

C++11

**operator<=>() [2/2]**

```
strong_ordering operator<=> (
    const error_condition & __lhs,
    const error_condition & __rhs) [related]
```

Ordered comparison for `std::error_condition`.

This defines a total order by comparing the categories, and then if they are equal comparing the values.

Since

C++11



**operator==(** [1/3]

```
bool operator== (
    const error_code & __lhs,
    const error_code & __rhs) [related]
```

Equality comparison for `std::error_code`.

Returns true only if they have the same category and the same value.

Since

C++11

**operator==(** [2/3]

```
bool operator== (
    const error_code & __lhs,
    const error_condition & __rhs) [related]
```

Equality comparison for `std::error_code` and `std::error_condition`.

Uses each category's `equivalent` member function to check whether the values correspond to an equivalent error in that category.

Since

C++11

**operator==(** [3/3]

```
bool operator== (
    const error_condition & __lhs,
    const error_condition & __rhs) [related]
```

Equality comparison for `std::error_condition`.

Returns true only if they have the same category and the same value.

Since

C++11

**system\_category()**

```
const error_category & std::system_category () [noexcept]
```

Error category for other error codes defined by the OS.

### 3.5.3 Exceptions

Collaboration diagram for Exceptions:



#### Classes

- class `__cxxabiv1::__forced_unwind`
- class `std::bad_alloc`
- class `std::experimental::fundamentals_v1::bad_any_cast`
- class `std::bad_cast`
- class `std::bad_exception`
- class `std::bad_function_call`
- class `std::experimental::fundamentals_v1::bad_optional_access`
- class `std::bad_typeid`
- class `std::bad_weak_ptr`
- class `std::domain_error`
- class `std::exception`
- class `std::__unspecified__::exception_ptr`
- struct `__gnu_cxx::forced_error`
- class `std::future_error`
- class `std::invalid_argument`
- class `std::length_error`
- class `std::logic_error`
- class `std::nested_exception`
- class `std::out_of_range`
- class `std::overflow_error`
- class `std::range_error`
- class `__gnu_cxx::recursive_init_error`
- class `std::regex_error`
- class `std::runtime_error`
- class `std::system_error`
- class `std::underflow_error`

## Typedefs

- typedef void(\* [std::terminate\\_handler](#)) ()
- typedef void(\* [std::unexpected\\_handler](#)) ()

## Functions

- void [\\_\\_gnu\\_cxx::\\_\\_verbose\\_terminate\\_handler](#) ()
- [exception\\_ptr](#) [std::current\\_exception](#) () noexcept
- [terminate\\_handler](#) [std::get\\_terminate](#) () noexcept
- [unexpected\\_handler](#) [std::get\\_unexpected](#) () noexcept
- template<typename \_Ex>  
  [exception\\_ptr](#) [std::make\\_exception\\_ptr](#) (\_Ex \_\_ex) noexcept
- void [std::rethrow\\_exception](#) ([exception\\_ptr](#))
- template<typename \_Ex>  
  void [std::rethrow\\_if\\_nested](#) (const \_Ex &\_\_ex)
- [terminate\\_handler](#) [std::set\\_terminate](#) ([terminate\\_handler](#)) noexcept
- [unexpected\\_handler](#) [std::set\\_unexpected](#) ([unexpected\\_handler](#)) noexcept
- void [std::terminate](#) () noexcept
- template<typename \_Tp>  
  void [std::throw\\_with\\_nested](#) (\_Tp &&\_\_t)
- bool [std::uncaught\\_exception](#) () noexcept
- void [std::unexpected](#) ()

## Variables

- class \_\_attribute\_\_((\_\_abi\_tag\_\_("cxx11"))) failure typedef \_los\_Fmtflags [std::ios\\_base::fmtflags](#)

### 3.5.3.1 Detailed Description

Since

C++98

Classes and functions for reporting errors via exceptions.

### 3.5.3.2 Typedef Documentation

#### **terminate\_handler**

```
typedef void(* std::terminate_handler) ()
```

If you write a replacement terminate handler, it must be of this type.

## unexpected\_handler

```
typedef void(* std::unexpected_handler) ()
```

If you write a replacement unexpected handler, it must be of this type.

### 3.5.3.3 Function Documentation

#### \_\_verbose\_terminate\_handler()

```
void __gnu_cxx::__verbose_terminate_handler ()
```

A replacement for the standard terminate\_handler which prints more information about the terminating exception (if any) on stderr.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler)
```

to use. For more info, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06s02.html>.

In 3.4 and later, this is on by default.

#### current\_exception()

```
exception_ptr std::current_exception () [noexcept]
```

Obtain an exception\_ptr to the currently handled exception.

If there is none, or the currently handled exception is foreign, return the null value.

Since

C++11

References [make\\_exception\\_ptr\(\)](#).

Referenced by [std::nested\\_exception::nested\\_exception\(\)](#), and [make\\_exception\\_ptr\(\)](#).

#### get\_terminate()

```
terminate_handler std::get_terminate () [noexcept]
```

Return the current terminate handler.

**get\_unexpected()**

```
unexpected_handler std::get_unexpected () [noexcept]
```

Return the current unexpected handler.

Since

C++11

**Deprecated** Removed from the C++ standard in C++17

**make\_exception\_ptr()**

```
template<typename _Ex>
exception_ptr std::make_exception_ptr (
    _Ex __ex) [noexcept]
```

Obtain an exception\_ptr pointing to a copy of the supplied object.

References [current\\_exception\(\)](#).

Referenced by [current\\_exception\(\)](#).

**rethrow\_exception()**

```
void std::rethrow_exception (
    exception_ptr )
```

Throw the object pointed to by the exception\_ptr.

References [rethrow\\_exception\(\)](#).

Referenced by [rethrow\\_exception\(\)](#), and [std::nested\\_exception::rethrow\\_nested\(\)](#).

**rethrow\_if\_nested()**

```
template<typename _Ex>
void std::rethrow_if_nested (
    const _Ex & __ex) [inline]
```

Rethrow a nested exception

If `__ex` contains a `std::nested_exception` object, call its `rethrow_nested()` member to rethrow the stored exception.

After catching an exception thrown by a call to `std::throw_with_nested` this function can be used to rethrow the exception that was active when `std::throw_with_nested` was called.

Since

C++11

**set\_terminate()**

```
terminate_handler std::set_terminate (
    terminate_handler ) [noexcept]
```

Takes a new handler function as an argument, returns the old function.

**set\_unexpected()**

```
unexpected_handler std::set_unexpected (
    unexpected_handler ) [noexcept]
```

Takes a new handler function as an argument, returns the old function.

**Deprecated** Removed from the C++ standard in C++17

**terminate()**

```
void std::terminate () [noexcept]
```

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

Referenced by [std::nested\\_exception::rethrow\\_nested\(\)](#).

**throw\_with\_nested()**

```
template<typename _Tp>
void std::throw_with_nested (
    _Tp && __t) [inline]
```

Throw an exception that also stores the currently active exception.

If `_Tp` is derived from `std::nested_exception` or is not usable as a base-class, throws a copy of `__t`. Otherwise, throws an object of an implementation-defined type derived from both `_Tp` and `std::nested_exception`, containing a copy of `__t` and the result of `std::current_exception()`.

In other words, throws the argument as a new exception that contains the currently active exception nested within it. This is intended for use in a catch handler to replace the caught exception with a different type, while still preserving the original exception. When the new exception is caught, the nested exception can be rethrown by using `std::rethrow_if_nested`.

This can be used at API boundaries, for example to catch a library's internal exception type and rethrow it nested with a `std::runtime_error`, or vice versa.

Since

C++11

References [forward\(\)](#).

## uncaught\_exception()

```
bool std::uncaught_exception () [noexcept]
```

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering `unexpected()` due to the throw; or after entering `terminate()` for any reason other than an explicit call to `terminate()`. [Note: This includes stack unwinding [15.2]. end note]'

2: 'When `uncaught_exception()` is true, throwing an exception can result in a call of `terminate()`' (15.5.1).'

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits >::sentry::~~sentry\(\)](#).

## unexpected()

```
void std::unexpected ()
```

The runtime will call this function if an exception is thrown which violates the function's exception specification.

**Deprecated** Removed from the C++ standard in C++17

### 3.5.3.4 Variable Documentation

#### fmtflags

```
class __attribute__((__abi_tag__ ("cxx11"))) failure typedef _Ios_Fmtflags std::ios_base::fmtflags
```

These are thrown to indicate problems with io.

##### 27.4.2.1.1 Class `ios_base::failure`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`

- [scientific](#)
- [showbase](#)
- [showpoint](#)
- [showpos](#)
- [skipws](#)
- [unitbuf](#)
- [uppercase](#)
- [adjustfield](#)
- [basefield](#)
- [floatfield](#)

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [register\\_callback\(\)](#), [setf\(\)](#), [setf\(\)](#), and [unsetf\(\)](#).

## 3.6 Extensions

Collaboration diagram for Extensions:



### Topics

- [Dynamic Bitset.](#)
- [Policy-Based Data Structures](#)
- [SGI](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std::tr2](#)



## Classes

- class `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`

### 3.6.1 Detailed Description

Components generally useful that are not part of any standard.

### 3.6.2 Dynamic Bitset.

Collaboration diagram for Dynamic Bitset.:



## Classes

- struct `std::tr2::__dynamic_bitset_base< _WordT, _Alloc >`
- class `std::tr2::dynamic_bitset< _WordT, _Alloc >`

## Functions

- template<typename \_CharT, typename \_Traits, typename \_Alloc1>  
void `std::tr2::dynamic_bitset< _WordT, _Alloc >::M_copy_to_string` (`std::basic_string< _CharT, _Traits, ↵  
_Alloc1 > &__str`, `_CharT __zero= _CharT('0')`, `_CharT __one= _CharT('1')`) const
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc>  
`std::basic_ostream< _CharT, _Traits > & std::tr2::operator<<` (`std::basic_ostream< _CharT, _Traits > &__os`,  
const `dynamic_bitset< _WordT, _Alloc > &__x`)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc>  
`std::basic_istream< _CharT, _Traits > & std::tr2::operator>>` (`std::basic_istream< _CharT, _Traits > &__↵  
is`, `dynamic_bitset< _WordT, _Alloc > &__x`)
- template<typename \_WordT, typename \_Alloc>  
bool `std::tr2::operator!=` (const `dynamic_bitset< _WordT, _Alloc > &__lhs`, const `dynamic_bitset< _WordT, _Alloc  
> &__rhs`)
- template<typename \_WordT, typename \_Alloc>  
bool `std::tr2::operator<=` (const `dynamic_bitset< _WordT, _Alloc > &__lhs`, const `dynamic_bitset< _WordT, _↵  
Alloc > &__rhs`)

- `template<typename _WordT, typename _Alloc>`  
`bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>`  
`bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`

### 3.6.2.1 Detailed Description

### 3.6.2.2 Function Documentation

#### **operator"!="()**

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator!= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

#### **operator&()**

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y) [inline]
```

Global bitwise operations on bitsets.

#### Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

#### Returns

A new bitset.

These should be self-explanatory.

**operator-()**

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y) [inline]
```

Global bitwise operations on bitsets.

**Parameters**

$\leftarrow$ __x	A bitset.
$\leftarrow$ __y	A bitset of the same size as __x.

**Returns**

A new bitset.

These should be self-explanatory.

**operator<<()**

```
template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const dynamic_bitset< _WordT, _Alloc > & __x) [inline]
```

Stream output operator for dynamic\_bitset.

**operator<=()**

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator<= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

**operator>()**

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator> (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

**operator>=()**

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator>= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

**operator>>()**

```
template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    dynamic_bitset< _WordT, _Alloc > & __x)
```

Stream input operator for dynamic\_bitset.

Input will skip whitespace and only accept '0' and '1' characters. The dynamic\_bitset will grow as necessary to hold the string of bits.

References `std::basic_string< _CharT, _Traits, _Alloc >::empty()`, `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, `std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_string< _CharT, _Traits, _Alloc >::size()`, `std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset()` and `std::basic_ios< _CharT, _Traits >::widen()`.

**operator^()**

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y) [inline]
```

Global bitwise operations on bitsets.

**Parameters**

$\_x$	A bitset.
$\_y$	A bitset of the same size as $\_x$ .

**Returns**

A new bitset.

These should be self-explanatory.

**operator" | ()**

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y) [inline]
```

Global bitwise operations on bitsets.

**Parameters**

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

**Returns**

A new bitset.

These should be self-explanatory.

**3.6.3 Policy-Based Data Structures**

Collaboration diagram for Policy-Based Data Structures:

**Topics**

- [Containers](#)
- [Exceptions](#)
- [Tags](#)
- [Traits](#)

**Classes**

- `struct \_\_gnu\_pbds::detail::container\_base\_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >`

### 3.6.3.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: [http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb\\_ds/index.html](http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html)

### 3.6.3.2 Containers

Collaboration diagram for Containers:



#### Topics

- [Branch-Based](#)
- [Hash-Based](#)
- [Heap-Based](#)
- [List-Based](#)

### 3.6.3.2.1 Detailed Description

### 3.6.3.2.2 Branch-Based

Collaboration diagram for Branch-Based:



## Topics

- [Base and Policy Classes](#)

## Classes

- class [\\_\\_gnu\\_pbds::basic\\_branch< Key, Mapped, Tag, Node\\_Update, Policy\\_Tl, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::tree< Key, Mapped, Cmp\\_Fn, Tag, Node\\_Update, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::trie< Key, Mapped, \\_ATraits, Tag, Node\\_Update, \\_Alloc >](#)

## Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

## Detailed Description

### Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



### \*Classes

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map< Key, Mapped, Node\\_And\\_It\\_Traits, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::splay\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#)

## Detailed Description

### 3.6.3.2.3 Hash-Based

Collaboration diagram for Hash-Based:



## Topics

- [Base and Policy Classes](#)

## Classes

- class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >`
- class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`
- class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`

## Macros

- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`

## Detailed Description

### Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



## \*Classes

- class `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`
- class `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`



## Detailed Description

### 3.6.3.2.4 Heap-Based

Collaboration diagram for Heap-Based:



## Topics

- [Base and Policy Classes](#)

## Classes

- [class `\_\_gnu\_pbds::priority\_queue<\_Tv, Cmp\_Fn, Tag, \_Alloc>`](#)

## Detailed Description

### Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



## \*Classes

- [class `\_\_gnu\_pbds::detail::binary\_heap<Value\_Type, Cmp\_Fn, \_Alloc>`](#)
- [class `\_\_gnu\_pbds::detail::binomial\_heap<Value\_Type, Cmp\_Fn, \_Alloc>`](#)
- [class `\_\_gnu\_pbds::detail::pairing\_heap<Value\_Type, Cmp\_Fn, \_Alloc>`](#)
- [class `\_\_gnu\_pbds::detail::rc\_binomial\_heap<Value\_Type, Cmp\_Fn, \_Alloc>`](#)
- [class `\_\_gnu\_pbds::detail::thin\_heap<Value\_Type, Cmp\_Fn, \_Alloc>`](#)

## Detailed Description

### 3.6.3.2.5 List-Based

Collaboration diagram for List-Based:



## Classes

- class [\\_\\_gnu\\_pbds::list\\_update< Key, Mapped, Eq\\_Fn, Update\\_Policy, \\_Alloc >](#)

## Macros

- `#define PB_DS_LU_BASE`

## Detailed Description

### 3.6.3.3 Exceptions

Collaboration diagram for Exceptions:



## Classes

- struct [\\_\\_gnu\\_pbds::container\\_error](#)
- struct [\\_\\_gnu\\_pbds::insert\\_error](#)
- struct [\\_\\_gnu\\_pbds::join\\_error](#)
- struct [\\_\\_gnu\\_pbds::resize\\_error](#)

## Functions

- void `__gnu_pbds::__throw_container_error()`
- void `__gnu_pbds::__throw_insert_error()`
- void `__gnu_pbds::__throw_join_error()`
- void `__gnu_pbds::__throw_resize_error()`

### 3.6.3.3.1 Detailed Description

### 3.6.3.4 Tags

Collaboration diagram for Tags:



## Topics

- [Data Structure Type](#)
- [Invalidation Guarantees](#)

## Classes

- struct `__gnu_pbds::trivial_iterator_tag`

## Typedefs

- typedef void `__gnu_pbds::trivial_iterator_difference_type`

### 3.6.3.4.1 Detailed Description

### 3.6.3.4.2 Typedef Documentation

#### `trivial_iterator_difference_type`

```
typedef void __gnu_pbds::trivial_iterator_difference_type
```

Prohibit moving trivial iterators.

### 3.6.3.4.3 Data Structure Type

Collaboration diagram for Data Structure Type:



#### Classes

- struct [\\_\\_gnu\\_pbds::associative\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_branch\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::binary\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::cc\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::container\\_tag](#)
- struct [\\_\\_gnu\\_pbds::gp\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::list\\_update\\_tag](#)
- struct [\\_\\_gnu\\_pbds::ov\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pairing\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pat\\_trie\\_tag](#)
- struct [\\_\\_gnu\\_pbds::priority\\_queue\\_tag](#)
- struct [\\_\\_gnu\\_pbds::rb\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::rc\\_binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::sequence\\_tag](#)
- struct [\\_\\_gnu\\_pbds::splay\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::string\\_tag](#)
- struct [\\_\\_gnu\\_pbds::thin\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trie\\_tag](#)

#### Detailed Description

### 3.6.3.4.4 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:



## Classes

- struct [\\_\\_gnu\\_pbds::basic\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::point\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::range\\_invalidation\\_guarantee](#)

## Detailed Description

### 3.6.3.5 Traits

Collaboration diagram for Traits:



## Classes

- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits< Cntr >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< \\_Tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< binary\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< binomial\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< cc\\_hash\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< gp\\_hash\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< list\\_update\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< ov\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< pairing\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< pat\\_trie\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< rb\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< splay\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< thin\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::detail::maybe\\_null\\_type< Key, Mapped, \\_Alloc, Store\\_Hash >](#)
- struct [\\_\\_gnu\\_pbds::detail::maybe\\_null\\_type< Key, null\\_type, \\_Alloc, Store\\_Hash >](#)
- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, Mapped >](#)
- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::null\\_node\\_update< \\_Tp1, \\_Tp2, \\_Tp3, \\_Tp4 >](#)
- struct [\\_\\_gnu\\_pbds::null\\_type](#)
- struct [\\_\\_gnu\\_pbds::detail::rebind\\_traits< \\_Alloc, T >](#)
- struct [\\_\\_gnu\\_pbds::detail::select\\_value\\_type< Key, Mapped >](#)

- struct `__gnu_pbds::detail::select_value_type< Key, null_type >`
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >`
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th, false >`
- struct `__gnu_pbds::detail::stored_hash< _Th >`
- struct `__gnu_pbds::detail::stored_value< _Tv >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`

## Variables

- template<typename Key, typename \_Alloc, bool Store\_Hash>  
`null_type __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >::s_null_type`

### 3.6.3.5.1 Detailed Description

## 3.6.4 SGI

Collaboration diagram for SGI:



## Classes

- class `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`
- struct `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`
- struct `__gnu_cxx::constant_unary_fun< _Result, _Argument >`
- struct `__gnu_cxx::constant_void_fun< _Result >`
- class `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`
- struct `__gnu_cxx::project1st< _Arg1, _Arg2 >`
- struct `__gnu_cxx::project2nd< _Arg1, _Arg2 >`
- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`
- class `__gnu_cxx::rope< _CharT, _Alloc >`
- struct `__gnu_cxx::select1st< _Pair >`
- struct `__gnu_cxx::select2nd< _Pair >`
- class `__gnu_cxx::slist< _Tp, _Alloc >`
- class `__gnu_cxx::subtractive_rng`
- struct `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`
- class `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`

## Functions

- template<typename \_Tp>  
const \_Tp & `__gnu_cxx::__median` (const \_Tp &\_\_a, const \_Tp &\_\_b, const \_Tp &\_\_c)
- template<typename \_Tp, typename \_Compare>  
const \_Tp & `__gnu_cxx::__median` (const \_Tp &\_\_a, const \_Tp &\_\_b, const \_Tp &\_\_c, \_Compare \_\_comp)
- constexpr size\_t `std::bitset< _Nb >::Find_first` () const noexcept
- constexpr size\_t `std::bitset< _Nb >::Find_next` (size\_t \_\_prev) const noexcept
- template<class \_Operation1, class \_Operation2>  
`unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1` (const \_Operation1 &\_\_fn1, const \_←  
\_Operation2 &\_\_fn2)
- template<class \_Operation1, class \_Operation2, class \_Operation3>  
`binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2` (const \_Operation1 &\_\_fn1,  
const \_Operation2 &\_\_fn2, const \_Operation3 &\_\_fn3)
- template<class \_Result>  
`constant_void_fun< _Result > __gnu_cxx::constant0` (const \_Result &\_\_val)
- template<class \_Result>  
`constant_unary_fun< _Result, _Result > __gnu_cxx::constant1` (const \_Result &\_\_val)
- template<class \_Result>  
`constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2` (const \_Result &\_\_val)
- template<typename \_InputIterator, typename \_Size, typename \_OutputIterator>  
`std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n` (\_InputIterator \_\_first, \_Size \_\_count, \_Output←  
Iterator \_\_result)
- template<typename \_InputIterator, typename \_Distance>  
void `__gnu_cxx::distance` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Distance &\_\_n)
- template<class \_Tp>  
\_Tp `__gnu_cxx::identity_element` (std::multiplies< \_Tp >)
- template<class \_Tp>  
\_Tp `__gnu_cxx::identity_element` (std::plus< \_Tp >)

- `template<typename _InputIterator1, typename _InputIterator2>`  
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp, typename _Integer>`  
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`  
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`  
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>`  
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>`  
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)`
- `constexpr bitset< _Nb > & std::bitset< _Nb >::Unchecked_set (size_t __pos) noexcept`
- `constexpr bitset< _Nb > & std::bitset< _Nb >::Unchecked_set (size_t __pos, int __val) noexcept`
- `constexpr bitset< _Nb > & std::bitset< _Nb >::Unchecked_reset (size_t __pos) noexcept`
- `constexpr bitset< _Nb > & std::bitset< _Nb >::Unchecked_flip (size_t __pos) noexcept`
- `constexpr bool std::bitset< _Nb >::Unchecked_test (size_t __pos) const noexcept`

### 3.6.4.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

**NB** that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function `compose2` takes `f`, `g1`, and `g2`, and constructs the `binary_compose` instance for you. For example, if `f` returns an `int`, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to



```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()`s take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()`s completely ignore any arguments passed, and return the stored value.

- `constant_void_fun`'s `operator()` takes no arguments
- `constant_unary_fun`'s `operator()` takes one argument (ignored)
- `constant_binary_fun`'s `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

### 3.6.4.2 Function Documentation

#### `__median()` [1/2]

```
template<typename _Tp>
const _Tp & __gnu_cxx::__median (
    const _Tp & __a,
    const _Tp & __b,
    const _Tp & __c)
```

Find the median of three values.

#### Parameters

<code>__a</code>	A value.
<code>__b</code>	A value.
<code>__c</code>	A value.

#### Returns

One of `a`, `b` or `c`.

If  $\{1,m,n\}$  is some convolution of  $\{a,b,c\}$  such that  $1 \leq m \leq n$  then the value returned will be  $m$ . This is an SGI extension.

#### `__median()` [2/2]

```
template<typename _Tp, typename _Compare>
const _Tp & __gnu_cxx::__median (
    const _Tp & __a,
    const _Tp & __b,
    const _Tp & __c,
    _Compare __comp)
```

Find the median of three values using a predicate for comparison.

**Parameters**

<code>__a</code>	A value.
<code>__b</code>	A value.
<code>__c</code>	A value.
<code>__comp</code>	A binary predicate.

**Returns**

One of `a`, `b` or `c`.

If  $\{l,m,n\}$  is some convolution of  $\{a,b,c\}$  such that `comp` (`l`, `m`) and `comp` (`m`, `n`) are both true then the value returned will be `m`. This is an SGI extension.

**`_Find_first()`**

```
template<size_t _Nb>
size_t std::bitset<_Nb>::_Find_first () const [inline], [constexpr], [noexcept]
Finds the index of the first "on" bit.
```

**Returns**

The index of the first bit set, or `size()` if not found.

**See also**

`_Find_next`

**`_Find_next()`**

```
template<size_t _Nb>
size_t std::bitset<_Nb>::_Find_next (
    size_t __prev) const [inline], [constexpr], [noexcept]
Finds the index of the next "on" bit after prev.
```

**Returns**

The index of the next bit set, or `size()` if not found.

**Parameters**

<code>__prev</code>	Where to start searching.
---------------------	---------------------------

**See also**

`_Find_first`

**`_Unchecked_flip()`**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::_Unchecked_flip (
    size_t __pos) [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

**\_Unchecked\_reset()**

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_reset (
    size_t __pos) [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

**\_Unchecked\_set() [1/2]**

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (
    size_t __pos) [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

**\_Unchecked\_set() [2/2]**

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (
    size_t __pos,
    int __val) [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

**\_Unchecked\_test()**

```
template<size_t _Nb>
bool std::bitset< _Nb >::_Unchecked_test (
    size_t __pos) const [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

**compose1()**

```
template<class _Operation1, class _Operation2>
unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (
    const _Operation1 & __fn1,
    const _Operation2 & __fn2) [inline]
```

An [SGI extension](#) .

**compose2()**

```
template<class _Operation1, class _Operation2, class _Operation3>
binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (
    const _Operation1 & __fn1,
    const _Operation2 & __fn2,
    const _Operation3 & __fn3) [inline]
```

An [SGI extension](#) .

**constant0()**

```
template<class _Result>
constant_void_fun< _Result > __gnu_cxx::constant0 (
    const _Result & __val) [inline]
```

An [SGI extension](#) .

**constant1()**

```
template<class _Result>
constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (
    const _Result & __val) [inline]
```

An [SGI extension](#) .

**constant2()**

```
template<class _Result>
constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (
    const _Result & __val) [inline]
```

An [SGI extension](#) .

**copy\_n()**

```
template<typename _InputIterator, typename _Size, typename _OutputIterator>
std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (
    _InputIterator __first,
    _Size __count,
    _OutputIterator __result) [inline]
```

Copies the range [first,first+count) into [result,result+count).

**Parameters**

<code>__first</code>	An input iterator.
<code>__count</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

**Returns**

A std::pair composed of first+count and result+count.

This is an SGI extension. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

**distance()**

```
template<typename _InputIterator, typename _Distance>
void __gnu_cxx::distance (
    _InputIterator __first,
    _InputIterator __last,
    _Distance & __n) [inline]
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

**identity\_element()** [1/2]

```
template<class _Tp>
_Tp __gnu_cxx::identity_element (
    std::multiplies< _Tp > ) [inline]
```

An [SGI extension](#) .

**identity\_element()** [2/2]

```
template<class _Tp>
_Tp __gnu_cxx::identity_element (
    std::plus< _Tp > ) [inline]
```

An SGI extension .

**lexicographical\_compare\_3way()**

```
template<typename _InputIterator1, typename _InputIterator2>
int __gnu_cxx::lexicographical_compare_3way (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2)
```

memcmp on steroids.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

**Returns**

An int, as with memcmp.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

**power()** [1/2]

```
template<typename _Tp, typename _Integer>
_Tp __gnu_cxx::power (
    _Tp __x,
    _Integer __n) [inline]
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

**power()** [2/2]

```
template<typename _Tp, typename _Integer, typename _MonoidOperation>
_Tp __gnu_cxx::power (
    _Tp __x,
    _Integer __n,
    _MonoidOperation __monoid_op) [inline]
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

**random\_sample()** [1/2]

```
template<typename _InputIterator, typename _RandomAccessIterator>
_RandomAccessIterator __gnu_cxx::random_sample (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __out_first,
    _RandomAccessIterator __out_last) [inline]
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

**random\_sample()** [2/2]

```
template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
_RandomAccessIterator __gnu_cxx::random_sample (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __out_first,
    _RandomAccessIterator __out_last,
    _RandomNumberGenerator & __rand) [inline]
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

**random\_sample\_n()** [1/2]

```
template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
_OutputIterator __gnu_cxx::random_sample_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _OutputIterator __out,
    const _Distance __n)
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

**random\_sample\_n()** [2/2]

```
template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename ↵
_RandomNumberGenerator>
_OutputIterator __gnu_cxx::random_sample_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _OutputIterator __out,
    const _Distance __n,
    _RandomNumberGenerator & __rand)
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

**uninitialized\_copy\_n()**

```
template<typename _InputIter, typename _Size, typename _ForwardIter>
std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (
    _InputIter __first,
    _Size __count,
    _ForwardIter __result) [inline]
```

Copies the range [first,last) into result.

**Parameters**

<code>__first</code>	An input iterator.
<code>__count</code>	Length
<code>__result</code>	An output iterator.

**Returns**

`__result + (__first + __count)`

Like `copy()`, but does not require an initialized output range.

**3.7 Filesystem****Files**

- file [filesystem](#)

**Classes**

- class [std::filesystem::directory\\_entry](#)
- class [std::filesystem::directory\\_iterator](#)
- class [std::filesystem::file\\_status](#)
- class [std::filesystem::filesystem\\_error](#)
- class [std::filesystem::path::iterator](#)
- class [std::filesystem::path](#)
- class [std::filesystem::recursive\\_directory\\_iterator](#)
- struct [std::filesystem::space\\_info](#)

**Typedefs**

- using [std::filesystem::file\\_time\\_type](#)

**Enumerations**

- enum class [std::filesystem::copy\\_options](#) : unsigned short { **none** , **skip\_existing** , **overwrite\_existing** , **update\_existing** , **recursive** , **copy\_symlinks** , **skip\_symlinks** , **directories\_only** , **create\_symlinks** , **create\_hard\_links** }
- enum class [std::filesystem::directory\\_options](#) : unsigned char { **none** , **follow\_directory\_symlink** , **skip\_permission\_denied** }
- enum class [std::filesystem::file\\_type](#) : signed char { **none** , **not\_found** , **regular** , **directory** , **symlink** , **block** , **character** , **fifo** , **socket** , **unknown** }



- enum class `std::filesystem::perm_options` : unsigned { **replace** , **add** , **remove** , **nofollow** }
- enum class `std::filesystem::perms` : unsigned {  
**none** , **owner\_read** , **owner\_write** , **owner\_exec** ,  
**owner\_all** , **group\_read** , **group\_write** , **group\_exec** ,  
**group\_all** , **others\_read** , **others\_write** , **others\_exec** ,  
**others\_all** , **all** , **set\_uid** , **set\_gid** ,  
**sticky\_bit** , **mask** , **unknown** }

## Functions

- `path std::filesystem::absolute` (const `path` &\_\_p)
- `path std::filesystem::absolute` (const `path` &\_\_p, `error_code` &\_\_ec)
- `path & std::filesystem::path::assign` (`string_type` &&\_\_source)
- `iterator std::filesystem::path::begin` () const noexcept
- `path std::filesystem::canonical` (const `path` &\_\_p)
- `path std::filesystem::canonical` (const `path` &\_\_p, `error_code` &\_\_ec)
- `int std::filesystem::path::compare` (const `string_type` &\_\_s) const noexcept
- `int std::filesystem::path::compare` (const `value_type` \*\_\_s) const noexcept
- `void std::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to)
- `void std::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options)
- `void std::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options, `error_code` &\_\_ec)
- `void std::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec)
- `bool std::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to)
- `bool std::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_option)
- `bool std::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_option, `error_code` &\_\_ec)
- `bool std::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec)
- `void std::filesystem::copy_symlink` (const `path` &\_\_existing\_symlink, const `path` &\_\_new\_symlink)
- `void std::filesystem::copy_symlink` (const `path` &\_\_existing\_symlink, const `path` &\_\_new\_symlink, `error_code` &\_\_ec) noexcept
- `bool std::filesystem::create_directories` (const `path` &\_\_p)
- `bool std::filesystem::create_directories` (const `path` &\_\_p, `error_code` &\_\_ec)
- `bool std::filesystem::create_directory` (const `path` &\_\_p)
- `bool std::filesystem::create_directory` (const `path` &\_\_p, const `path` &\_\_attributes)
- `bool std::filesystem::create_directory` (const `path` &\_\_p, const `path` &\_\_attributes, `error_code` &\_\_ec) noexcept
- `bool std::filesystem::create_directory` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `void std::filesystem::create_directory_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink)
- `void std::filesystem::create_directory_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink, `error_code` &\_\_ec) noexcept
- `void std::filesystem::create_hard_link` (const `path` &\_\_to, const `path` &\_\_new\_hard\_link)
- `void std::filesystem::create_hard_link` (const `path` &\_\_to, const `path` &\_\_new\_hard\_link, `error_code` &\_\_ec) noexcept
- `void std::filesystem::create_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink)
- `void std::filesystem::create_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink, `error_code` &\_\_ec) noexcept
- `path std::filesystem::current_path` ()
- `void std::filesystem::current_path` (const `path` &\_\_p)
- `void std::filesystem::current_path` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `path std::filesystem::current_path` (`error_code` &\_\_ec)
- `iterator std::filesystem::path::end` () const noexcept
- `bool std::filesystem::equivalent` (const `path` &\_\_p1, const `path` &\_\_p2)
- `bool std::filesystem::equivalent` (const `path` &\_\_p1, const `path` &\_\_p2, `error_code` &\_\_ec) noexcept
- `bool std::filesystem::exists` (const `path` &\_\_p)

- `bool std::filesystem::exists (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::exists (file_status) noexcept`
- `path std::filesystem::path::extension () const`
- `uintmax_t std::filesystem::file_size (const path &)`
- `uintmax_t std::filesystem::file_size (const path &, error_code &) noexcept`
- `path std::filesystem::path::filename () const`
- `std::string std::filesystem::path::generic_string () const`
- `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>  
std::basic_string<_CharT, _Traits, _Allocator> std::filesystem::path::generic_string (const _Allocator &__a= _Allocator()) const`
- `std::u16string std::filesystem::path::generic_u16string () const`
- `std::u32string std::filesystem::path::generic_u32string () const`
- `std::string std::filesystem::path::generic_u8string () const`
- `std::wstring std::filesystem::path::generic_wstring () const`
- `uintmax_t std::filesystem::hard_link_count (const path &)`
- `uintmax_t std::filesystem::hard_link_count (const path &, error_code &) noexcept`
- `bool std::filesystem::path::has_extension () const noexcept`
- `bool std::filesystem::path::has_stem () const noexcept`
- `size_t std::filesystem::hash_value (const path &__p) noexcept`
- `bool std::filesystem::path::is_absolute () const noexcept`
- `bool std::filesystem::is_block_file (const path &__p)`
- `bool std::filesystem::is_block_file (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_block_file (file_status __s) noexcept`
- `bool std::filesystem::is_character_file (const path &__p)`
- `bool std::filesystem::is_character_file (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_character_file (file_status __s) noexcept`
- `bool std::filesystem::is_directory (const path &__p)`
- `bool std::filesystem::is_directory (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_directory (file_status __s) noexcept`
- `bool std::filesystem::is_empty (const path &__p)`
- `bool std::filesystem::is_empty (const path &__p, error_code &__ec)`
- `bool std::filesystem::is_fifo (const path &__p)`
- `bool std::filesystem::is_fifo (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_fifo (file_status __s) noexcept`
- `bool std::filesystem::is_other (const path &__p)`
- `bool std::filesystem::is_other (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_other (file_status) noexcept`
- `bool std::filesystem::is_regular_file (const path &__p)`
- `bool std::filesystem::is_regular_file (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_regular_file (file_status) noexcept`
- `bool std::filesystem::is_socket (const path &__p)`
- `bool std::filesystem::is_socket (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_socket (file_status __s) noexcept`
- `bool std::filesystem::is_symlink (const path &__p)`
- `bool std::filesystem::is_symlink (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_symlink (file_status) noexcept`
- `file_time_type std::filesystem::last_write_time (const path &)`
- `file_time_type std::filesystem::last_write_time (const path &, error_code &) noexcept`
- `void std::filesystem::last_write_time (const path &__p, file_time_type __new_time)`
- `void std::filesystem::last_write_time (const path &__p, file_time_type __new_time, error_code &__ec) noexcept`

- `path & std::filesystem::path::make_preferred ()`
- `copy_options & std::filesystem::operator+= (copy_options &__x, copy_options __y) noexcept`
- `reference std::filesystem::path::iterator::operator* () const noexcept`
- `iterator & std::filesystem::path::iterator::operator++ () noexcept`
- `template<typename _CharT>  
__detail::_Path2< _CharT * > & std::filesystem::path::operator+= (_CharT __x)`
- `path & std::filesystem::path::operator+= (basic_string_view< value_type > __x)`
- `path & std::filesystem::path::operator+= (const string_type &__x)`
- `path & std::filesystem::path::operator+= (const value_type *__x)`
- `path & std::filesystem::path::operator+= (value_type __x)`
- `iterator & std::filesystem::path::iterator::operator-- () noexcept`
- `path & std::filesystem::path::operator= (path &&) noexcept`
- `path & std::filesystem::path::operator= (string_type && __source)`
- `constexpr copy_options std::filesystem::operator^ (copy_options __x, copy_options __y) noexcept`
- `copy_options & std::filesystem::operator^= (copy_options &__x, copy_options __y) noexcept`
- `constexpr copy_options std::filesystem::operator| (copy_options __x, copy_options __y) noexcept`
- `copy_options & std::filesystem::operator|= (copy_options &__x, copy_options __y) noexcept`
- `constexpr copy_options std::filesystem::operator~ (copy_options __x) noexcept`
- `void std::filesystem::permissions (const path &, perms, perm_options, error_code &) noexcept`
- `void std::filesystem::permissions (const path & __p, perms __prms, error_code & __ec) noexcept`
- `void std::filesystem::permissions (const path & __p, perms __prms, perm_options __opts=perm_options::replace)`
- `path std::filesystem::proximate (const path & __p, const path & __base, error_code & __ec)`
- `path std::filesystem::proximate (const path & __p, const path & __base=current_path())`
- `path std::filesystem::proximate (const path & __p, error_code & __ec)`
- `path std::filesystem::read_symlink (const path & __p)`
- `path std::filesystem::read_symlink (const path & __p, error_code & __ec)`
- `path std::filesystem::relative (const path & __p, const path & __base, error_code & __ec)`
- `path std::filesystem::relative (const path & __p, const path & __base=current_path())`
- `path std::filesystem::relative (const path & __p, error_code & __ec)`
- `bool std::filesystem::remove (const path &, error_code &) noexcept`
- `bool std::filesystem::remove (const path & __p)`
- `uintmax_t std::filesystem::remove_all (const path &)`
- `uintmax_t std::filesystem::remove_all (const path &, error_code &)`
- `void std::filesystem::rename (const path & __from, const path & __to)`
- `void std::filesystem::rename (const path & __from, const path & __to, error_code & __ec) noexcept`
- `void std::filesystem::resize_file (const path & __p, uintmax_t __size)`
- `void std::filesystem::resize_file (const path & __p, uintmax_t __size, error_code & __ec) noexcept`
- `space_info std::filesystem::space (const path & __p)`
- `space_info std::filesystem::space (const path & __p, error_code & __ec) noexcept`
- `file_status std::filesystem::status (const path &)`
- `file_status std::filesystem::status (const path &, error_code &) noexcept`
- `bool std::filesystem::status_known (file_status) noexcept`
- `path std::filesystem::path::stem () const`
- `std::string std::filesystem::path::string () const`
- `template<typename _CharT, typename _Traits, typename _Allocator>  
basic_string< _CharT, _Traits, _Allocator > std::filesystem::path::string (const _Allocator & __a) const`
- `void swap (path & __lhs, path & __rhs) noexcept`
- `void std::filesystem::path::swap (path & __rhs) noexcept`
- `file_status std::filesystem::symlink_status (const path &)`
- `file_status std::filesystem::symlink_status (const path &, error_code &) noexcept`

- `path std::filesystem::temp_directory_path ()`
- `path std::filesystem::temp_directory_path (error_code &__ec)`
- `std::u16string std::filesystem::path::u16string () const`
- `std::u32string std::filesystem::path::u32string () const`
- `template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _CharT = __detail::_value_type_<is_char_or_char8_t<_InputIterator>>>`  
`path u8path (_InputIterator __first, _InputIterator __last)`
- `template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __detail::_value_type_is_char_or_<_char8_t<_Source>>`  
`path u8path (const _Source &__source)`
- `std::string std::filesystem::path::u8string () const`
- `path std::filesystem::weakly_canonical (const path &__p)`
- `path std::filesystem::weakly_canonical (const path &__p, error_code &__ec)`
- `std::wstring std::filesystem::path::wstring () const`
- `constexpr perms std::filesystem::operator| (perms __x, perms __y) noexcept`
- `constexpr perms std::filesystem::operator^ (perms __x, perms __y) noexcept`
- `constexpr perms std::filesystem::operator~ (perms __x) noexcept`
- `perms & std::filesystem::operator&= (perms &__x, perms __y) noexcept`
- `perms & std::filesystem::operator|= (perms &__x, perms __y) noexcept`
- `perms & std::filesystem::operator^= (perms &__x, perms __y) noexcept`
- `constexpr perm_options std::filesystem::operator| (perm_options __x, perm_options __y) noexcept`
- `constexpr perm_options std::filesystem::operator^ (perm_options __x, perm_options __y) noexcept`
- `constexpr perm_options std::filesystem::operator~ (perm_options __x) noexcept`
- `perm_options & std::filesystem::operator&= (perm_options &__x, perm_options __y) noexcept`
- `perm_options & std::filesystem::operator|= (perm_options &__x, perm_options __y) noexcept`
- `perm_options & std::filesystem::operator^= (perm_options &__x, perm_options __y) noexcept`
- `constexpr directory_options std::filesystem::operator| (directory_options __x, directory_options __y) noexcept`
- `constexpr directory_options std::filesystem::operator^ (directory_options __x, directory_options __y) noexcept`
- `constexpr directory_options std::filesystem::operator~ (directory_options __x) noexcept`
- `directory_options & std::filesystem::operator&= (directory_options &__x, directory_options __y) noexcept`
- `directory_options & std::filesystem::operator|= (directory_options &__x, directory_options __y) noexcept`
- `directory_options & std::filesystem::operator^= (directory_options &__x, directory_options __y) noexcept`
- `directory_iterator begin (directory_iterator __iter) noexcept`
- `directory_iterator end (directory_iterator) noexcept`
- `recursive_directory_iterator begin (recursive_directory_iterator __iter) noexcept`
- `recursive_directory_iterator end (recursive_directory_iterator) noexcept`

### 3.7.1 Detailed Description

### 3.7.2 Typedef Documentation

#### file\_time\_type

using `std::filesystem::file_time_type`

The type used for file timestamps.

### 3.7.3 Enumeration Type Documentation

#### copy\_options

enum class `std::filesystem::copy_options` : unsigned short [strong]

Bitmask type controlling effects of `filesystem::copy`

### directory\_options

enum class `std::filesystem::directory_options` : unsigned char [strong]  
Bitmask type controlling directory iteration.

### file\_type

enum class `std::filesystem::file_type` : signed char [strong]  
Enumerated type representing the type of a file.

### perm\_options

enum class `std::filesystem::perm_options` : unsigned [strong]  
Bitmask type controlling changes to permissions.

### perms

enum class `std::filesystem::perms` : unsigned [strong]  
Bitmask type representing file access permissions.

## 3.7.4 Function Documentation

### begin() [1/2]

```
directory_iterator begin (  
    directory_iterator __iter) [related]
```

Enable range-based for using `directory_iterator`.

Since

C++17

e.g. `for (auto& entry : std::filesystem::directory_iterator(".")) ...`

### begin() [2/2]

```
recursive_directory_iterator begin (  
    recursive_directory_iterator __iter) [related]
```

Enable range-based for using `recursive_directory_iterator`.

Since

C++17

e.g. `for (auto& entry : recursive_directory_iterator(".")) ...`

### end() [1/2]

```
directory_iterator end (  
    directory_iterator ) [related]
```

Return a past-the-end `directory_iterator`.

### end() [2/2]

```
recursive_directory_iterator end (  
    recursive_directory_iterator ) [related]
```

Return a past-the-end `recursive_directory_iterator`.

**u8path()** [1/2]

```
template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename
_CharT = __detail::__value_type_is_char_or_char8_t<_InputIterator>>
path u8path (
    _InputIterator __first,
    _InputIterator __last) [related]
```

Create a path from a UTF-8-encoded sequence of char

Since

C++17

References [u8path\(\)](#).

Referenced by [u8path\(\)](#), and [u8path\(\)](#).

**u8path()** [2/2]

```
template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __↵
detail::__value_type_is_char_or_char8_t<_Source>>
path u8path (
    const _Source & __source) [related]
```

Create a path from a UTF-8-encoded sequence of char

Since

C++17

References [u8path\(\)](#).

**3.8 I/O**

Collaboration diagram for I/O:

**Classes**

- class [std::basic\\_filebuf<\\_CharT, \\_Traits>](#)
- class [std::basic\\_fstream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ifstream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ios<\\_CharT, \\_Traits>](#)
- class [std::basic\\_iostream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_istream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_istreamstream<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_ofstream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ostream<\\_CharT, \\_Traits>](#)

- class `std::basic_ostringstream<_CharT, _Traits, _Alloc>`
- class `std::basic_streambuf<_CharT, _Traits>`
- class `std::basic_stringbuf<_CharT, _Traits, _Alloc>`
- class `std::basic_stringstream<_CharT, _Traits, _Alloc>`
- class `std::ios_base`
- class `std::istreambuf_iterator<_CharT, _Traits>`
- class `std::ostreambuf_iterator<_CharT, _Traits>`
- class `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`
- class `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`

## Typedefs

- typedef `basic_filebuf<char>` `std::filebuf`
- typedef `basic_fstream<char>` `std::fstream`
- typedef `basic_ifstream<char>` `std::ifstream`
- typedef `basic_ios<char>` `std::ios`
- typedef `basic_iostream<char>` `std::iostream`
- typedef `basic_istream<char>` `std::istream`
- typedef `basic_istreamstream<char>` `std::istreamstream`
- typedef `basic_ofstream<char>` `std::ofstream`
- typedef `basic_ostream<char>` `std::ostream`
- typedef `basic_ostringstream<char>` `std::ostringstream`
- typedef `basic_streambuf<char>` `std::streambuf`
- typedef `basic_stringbuf<char>` `std::stringbuf`
- typedef `basic_stringstream<char>` `std::stringstream`
- typedef `basic_filebuf<wchar_t>` `std::wfilebuf`
- typedef `basic_fstream<wchar_t>` `std::wfstream`
- typedef `basic_ifstream<wchar_t>` `std::wifstream`
- typedef `basic_ios<wchar_t>` `std::wios`
- typedef `basic_iostream<wchar_t>` `std::wiostream`
- typedef `basic_istream<wchar_t>` `std::wistream`
- typedef `basic_istreamstream<wchar_t>` `std::wistreamstream`
- typedef `basic_ofstream<wchar_t>` `std::wofstream`
- typedef `basic_ostream<wchar_t>` `std::wostream`
- typedef `basic_ostringstream<wchar_t>` `std::wostringstream`
- typedef `basic_streambuf<wchar_t>` `std::wstreambuf`
- typedef `basic_stringbuf<wchar_t>` `std::wstringbuf`
- typedef `basic_stringstream<wchar_t>` `std::wstringstream`

### 3.8.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char> ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html#std.io.objects>

### 3.8.2 Typedef Documentation

#### filebuf

```
typedef basic_filebuf<char> std::filebuf
```

Class for `char` file buffers.

#### fstream

```
typedef basic_fstream<char> std::fstream
```

Class for `char` mixed input and output file streams.

#### ifstream

```
typedef basic_ifstream<char> std::ifstream
```

Class for `char` input file streams.

#### ios

```
typedef basic_ios<char> std::ios
```

Base class for `char` streams.

#### iostream

```
typedef basic_iostream<char> std::iostream
```

Base class for `char` mixed input and output streams.

#### istream

```
typedef basic_istream<char> std::istream
```

Base class for `char` input streams.

#### istringstream

```
typedef basic_istringstream<char> std::istringstream
```

Class for `char` input memory streams.

#### ofstream

```
typedef basic_ofstream<char> std::ofstream
```

Class for `char` output file streams.

#### ostream

```
typedef basic_ostream<char> std::ostream
```

Base class for `char` output streams.

#### ostringstream

```
typedef basic_ostringstream<char> std::ostringstream
```

Class for `char` output memory streams.

#### streambuf

```
typedef basic_streambuf<char> std::streambuf
```

Base class for `char` buffers.



**stringbuf**

typedef `basic_stringbuf<char>` `std::stringbuf`  
Class for `char` memory buffers.

**stringstream**

typedef `basic_stringstream<char>` `std::stringstream`  
Class for `char` mixed input and output memory streams.

**wfilebuf**

typedef `basic_filebuf<wchar_t>` `std::wfilebuf`  
Class for `wchar_t` file buffers.

**wfstream**

typedef `basic_fstream<wchar_t>` `std::wfstream`  
Class for `wchar_t` mixed input and output file streams.

**wifstream**

typedef `basic_ifstream<wchar_t>` `std::wifstream`  
Class for `wchar_t` input file streams.

**wios**

typedef `basic_ios<wchar_t>` `std::wios`  
Base class for `wchar_t` streams.

**wiostream**

typedef `basic_iostream<wchar_t>` `std::wiostream`  
Base class for `wchar_t` mixed input and output streams.

**wistream**

typedef `basic_istream<wchar_t>` `std::wistream`  
Base class for `wchar_t` input streams.

**wistringstream**

typedef `basic_istringstream<wchar_t>` `std::wistringstream`  
Class for `wchar_t` input memory streams.

**wofstream**

typedef `basic_ofstream<wchar_t>` `std::wofstream`  
Class for `wchar_t` output file streams.

**wostream**

typedef `basic_ostream<wchar_t>` `std::wostream`  
Base class for `wchar_t` output streams.

**wostreamstream**

typedef `basic_ostringstream<wchar_t>` `std::wostreamstream`  
 Class for `wchar_t` output memory streams.

**wstreambuf**

typedef `basic_streambuf<wchar_t>` `std::wstreambuf`  
 Base class for `wchar_t` buffers.

**wstringbuf**

typedef `basic_stringbuf<wchar_t>` `std::wstringbuf`  
 Class for `wchar_t` memory buffers.

**wstringstream**

typedef `basic_stringstream<wchar_t>` `std::wstringstream`  
 Class for `wchar_t` mixed input and output memory streams.

**3.9 Iterators**

Collaboration diagram for Iterators:

**Topics**

- [Iterator Tags](#)

**Classes**

- class `std::back_insert_iterator<_Container>`
- class `std::common_iterator<_It, _Sent>`
- class `std::counted_iterator<_It>`
- class `std::front_insert_iterator<_Container>`
- struct `std::input_iterator_tag`
- class `std::insert_iterator<_Container>`
- class `std::istream_iterator<_Tp, _CharT, _Traits, _Dist>`
- class `std::istreambuf_iterator<_CharT, _Traits>`
- struct `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>`

- struct `std::iterator_traits<_Iterator>`
- struct `std::iterator_traits<_Tp*>`
- class `std::move_iterator<_Iterator>`
- class `std::move_sentinel<_Sent>`
- class `std::ostream_iterator<_Tp, _CharT, _Traits>`
- class `std::ostreambuf_iterator<_CharT, _Traits>`
- class `std::reverse_iterator<_Iterator>`

## Macros

- `#define _GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR(_Iter)`
- `#define _GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

## Functions

- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT>>::__type std::__`  
`copy_move_a2(_CharT* __first, _CharT* __last, ostreambuf_iterator<_CharT> __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT>>::__type std::__`  
`copy_move_a2(const _CharT* __first, const _CharT* __last, ostreambuf_iterator<_CharT> __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT*>::__type std::__copy_move_a2`  
`(istreambuf_iterator<_CharT> __first, istreambuf_iterator<_CharT> __last, _CharT* __result)`
- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT*>::__type std::__copy_n_a(istreambuf_iterator<`  
`_CharT> __it, _Size __n, _CharT* __result, bool __strict)`
- `template<typename _Iter>`  
`constexpr iterator_traits<_Iter>::iterator_category std::__iterator_category(const _Iter &)`
- `template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<typename iterator_traits<_`  
`Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>`  
`constexpr _ReturnType std::__make_move_if_noexcept_iterator(_Iterator __i)`
- `template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_`  
`iterator<_Tp*>>>`  
`constexpr _ReturnType std::__make_move_if_noexcept_iterator(_Tp* __i)`
- `template<typename _Iterator>`  
`constexpr reverse_iterator<_Iterator> std::__make_reverse_iterator(_Iterator __i)`
- `template<typename _CharT, typename _Distance>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, void>::__type std::__advance(istreambuf_iterator<`  
`_CharT> & __i, _Distance __n)`
- `template<typename _Container>`  
`constexpr back_insert_iterator<_Container> std::back_inserter(_Container & __x)`
- `template<typename _CharT>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT>>::__type std::copy`  
`(istreambuf_iterator<_CharT> __first, istreambuf_iterator<_CharT> __last, ostreambuf_iterator<_CharT>`  
`__result)`
- `template<typename _CharT>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, istreambuf_iterator<_CharT>>::__type std::find`  
`(istreambuf_iterator<_CharT> __first, istreambuf_iterator<_CharT> __last, const _CharT & __val)`
- `template<typename _Container>`  
`constexpr front_insert_iterator<_Container> std::front_inserter(_Container & __x)`
- `template<typename _Container>`  
`constexpr insert_iterator<_Container> std::inserter(_Container & __x, std::__detail::__range_iter_t<_`  
`Container> __i)`

- `template<typename _Iterator>`  
`constexpr move\_iterator< _Iterator > std::make_move_iterator ( _Iterator __i)`
- `template<typename _Iterator>`  
`constexpr reverse\_iterator< _Iterator > std::make_reverse_iterator ( _Iterator __i)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator!= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator>`  
`requires requires { { __x.base() + __n } -> same_as< _Iterator>; }`  
`constexpr move\_iterator< _Iterator > std::operator+ (typename move\_iterator< _Iterator >::difference_type __n, const move\_iterator< _Iterator > &__x)`
- `template<typename _Iterator>`  
`constexpr reverse\_iterator< _Iterator > std::operator+ (typename reverse\_iterator< _Iterator >::difference_type __n, const reverse\_iterator< _Iterator > &__x)`
- `template<typename _IteratorL, typename _IteratorR>`  
`constexpr auto std::operator- (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y) -> decltype(__x.base() - __y.base())`
- `template<typename _IteratorL, typename _IteratorR>`  
`constexpr auto std::operator- (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y) -> decltype(__y.base() - __x.base())`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator< (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator<= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator<= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare\_three\_way\_result\_t< _Iterator > std::operator<= (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare\_three\_way\_result\_t< _IteratorL, _IteratorR > std::operator<= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare\_three\_way\_result\_t< _Iterator, _Iterator > std::operator<= (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare\_three\_way\_result\_t< _IteratorL, _IteratorR > std::operator<= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _CharT, typename _Traits>`  
`bool std::operator== (const istreambuf\_iterator< _CharT, _Traits > &__a, const istreambuf\_iterator< _CharT, _Traits > &__b)`



`std::vector<_Tp, polymorphic_allocator<_Tp>>::assign()`, `distance()`, `fill_n()`, `find_end()`, `find_end()`, `std::deque<_Tp, polymorphic_allocator<_Tp>>::insert()`, `partition()`, `reverse()`, `rotate()`, `uninitialized_copy_n()`, `unique_copy()`, and `unique_copy()`.

### back\_inserter()

```
template<typename _Container>
back_insert_iterator<_Container> std::back_inserter (
    _Container & __x) [inline], [nodiscard], [constexpr]
```

#### Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

#### Returns

An instance of `back_insert_iterator` working on `__x`.

This wrapper function helps in creating `back_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, `regex_replace()`, `regex_replace()`, `regex_replace()`, and `regex_replace()`.

### front\_inserter()

```
template<typename _Container>
front_insert_iterator<_Container> std::front_inserter (
    _Container & __x) [inline], [nodiscard], [constexpr]
```

#### Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

#### Returns

An instance of `front_insert_iterator` working on `x`.

This wrapper function helps in creating `front_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

### inserter()

```
template<typename _Container>
insert_iterator<_Container> std::inserter (
    _Container & __x,
    std::__detail::__range_iter_t<_Container> __i) [nodiscard], [constexpr]
```

**Parameters**

$\_x$	A container of arbitrary type.
$\_i$	An iterator into the container.

**Returns**

An instance of `insert_iterator` working on `__x`.

This wrapper function helps in creating `insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

**make\_reverse\_iterator()**

```
template<typename _Iterator>
reverse_iterator< _Iterator > std::make_reverse_iterator (
    _Iterator __i) [inline], [constexpr]
```

Generator function for `reverse_iterator`.

**operator==()**

```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
bool std::operator== (
    const reverse_iterator< _IteratorL > & __x,
    const reverse_iterator< _IteratorR > & __y) [nodiscard], [constexpr]
```

**Parameters**

$\_x$	A <code>reverse_iterator</code> .
$\_y$	A <code>reverse_iterator</code> .

**Returns**

A simple bool.

Reverse iterators forward comparisons to their underlying base() iterators.

References [std::reverse\\_iterator<\\_Iterator>::base\(\)](#).

**3.9.3 Iterator Tags**

Collaboration diagram for Iterator Tags:

**Classes**

- struct [std::bidirectional\\_iterator\\_tag](#)
- struct [std::contiguous\\_iterator\\_tag](#)
- struct [std::forward\\_iterator\\_tag](#)
- struct [std::input\\_iterator\\_tag](#)
- struct [std::output\\_iterator\\_tag](#)
- struct [std::random\\_access\\_iterator\\_tag](#)

**3.9.3.1 Detailed Description**

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

**3.10 Locales****Namespaces**

- namespace [std::\\_\\_detail](#)

**Classes**

- class [std::codecvt<\\_InternT, \\_ExternT, \\_StateT>](#)
- class [std::ctype<\\_CharT>](#)
- class [std::ctype<char>](#)
- class [std::ctype<wchar\\_t>](#)
- class [std::locale::facet](#)
- class [std::locale::id](#)
- class [std::locale](#)
- class [std::messages<\\_CharT>](#)
- struct [std::messages\\_base](#)
- class [std::money\\_base](#)



- class `std::money_get<_CharT, _Inlter >`
- class `std::money_put<_CharT, _Outlter >`
- class `std::moneypunct<_CharT, _Intl >`
- class `std::num_get<_CharT, _Inlter >`
- class `std::num_put<_CharT, _Outlter >`
- class `std::numpunct<_CharT >`
- class `std::time_base`
- class `std::time_get<_CharT, _Inlter >`
- class `std::time_put<_CharT, _Outlter >`
- class `std::wbuffer_convert<_Codecvt, _Elem, _Tr >`
- class `std::wstring_convert<_Codecvt, _Elem, _Wide_alloc, _Byte_alloc >`

## Functions

- template<typename \_OutStr, typename \_InChar, typename \_Codecvt, typename \_State, typename \_Fn>  
bool **std::do\_str\_codecvt** (const \_InChar \* \_\_first, const \_InChar \* \_\_last, \_OutStr & \_\_outstr, const \_Codecvt & \_\_cvt, \_State & \_\_state, size\_t & \_\_count, \_Fn \_\_fn)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, typename \_State>  
bool **std::str\_codecvt\_in** (const char \* \_\_first, const char \* \_\_last, [basic\\_string](#)<\_CharT, \_Traits, \_Alloc > & \_\_outstr, const [codecvt](#)<\_CharT, char, \_State > & \_\_cvt)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, typename \_State>  
bool **std::str\_codecvt\_in** (const char \* \_\_first, const char \* \_\_last, [basic\\_string](#)<\_CharT, \_Traits, \_Alloc > & \_\_outstr, const [codecvt](#)<\_CharT, char, \_State > & \_\_cvt, \_State & \_\_state, size\_t & \_\_count)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, typename \_State>  
bool **std::str\_codecvt\_in\_all** (const char \* \_\_first, const char \* \_\_last, [basic\\_string](#)<\_CharT, \_Traits, \_Alloc > & \_\_outstr, const [codecvt](#)<\_CharT, char, \_State > & \_\_cvt)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, typename \_State>  
bool **std::str\_codecvt\_out** (const \_CharT \* \_\_first, const \_CharT \* \_\_last, [basic\\_string](#)<char, \_Traits, \_Alloc > & \_\_outstr, const [codecvt](#)<\_CharT, char, \_State > & \_\_cvt)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, typename \_State>  
bool **std::str\_codecvt\_out** (const \_CharT \* \_\_first, const \_CharT \* \_\_last, [basic\\_string](#)<char, \_Traits, \_Alloc > & \_\_outstr, const [codecvt](#)<\_CharT, char, \_State > & \_\_cvt, \_State & \_\_state, size\_t & \_\_count)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, typename \_State>  
bool **std::str\_codecvt\_out\_all** (const \_CharT \* \_\_first, const \_CharT \* \_\_last, [basic\\_string](#)<char, \_Traits, \_Alloc > & \_\_outstr, const [codecvt](#)<\_CharT, char, \_State > & \_\_cvt)
- template<typename \_Facet>  
bool **std::has\_facet** (const [locale](#) & \_\_loc)
- template<typename \_Facet>  
const \_Facet & **std::use\_facet** (const [locale](#) & \_\_loc)

### 3.10.1 Detailed Description

Classes and functions for internationalization and localization.

### 3.10.2 Function Documentation

#### **has\_facet()**

```
template<typename _Facet>
bool std::has_facet (
    const locale & __loc) [inline], [nodiscard], [noexcept]
```

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

## Template Parameters

<code>_Facet</code>	The facet type to test the presence of.
---------------------	---

## Parameters

<code>__loc</code>	The locale to test.
--------------------	---------------------

## Returns

true if `__loc` contains a facet of type `_Facet`, else false.

**use\_facet()**

```
template<typename _Facet>
const _Facet & std::use_facet (
    const locale & __loc) [inline], [nodiscard]
```

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

## Template Parameters

<code>_Facet</code>	The facet type to access.
---------------------	---------------------------

## Parameters

<code>__loc</code>	The locale to use.
--------------------	--------------------

## Returns

Reference to facet of type `Facet`.

## Exceptions

<code>std::bad_cast</code>	if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> .
----------------------------	---

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::time_get<_CharT, _InIter>::get()`, `isalnum()`, `isalpha()`, `isblank()`, `iscntrl()`, `std::regex_traits<_Ch_type>::isctype()`, `isdigit()`, `isgraph()`, `islower()`, `isprint()`, `ispunct()`, `isspace()`, `isupper()`, `isxdigit()`, `std::regex_traits<_Ch_type>::lookup_classname()`, `std::regex_traits<_CharT>::lookup_classname()`, `std::regex_traits<_Ch_type>::lookup_collatename()`, `std::basic_ostream<_CharT, _Traits>::operator<>()`, `std::basic_istream<_CharT, _Traits>::operator<>()`, `operator<>()`, `std::time_put<_CharT, _OutIter>::put()`, `tolower()`, `toupper()`, `std::regex_traits<_CharT>::transform()`, `std::regex_traits<_CharT>::transform_primary()`, `std::regex_traits<_CharT>::translate_nocase()`, and `ws()`.

### 3.11 Numerics

Collaboration diagram for Numerics:



#### Topics

- [Bit manipulation](#)
- [Complex Numbers](#)
- [Decimal Floating-Point Arithmetic](#)
- [Mathematical Special Functions](#)
- [Numeric Arrays](#)
- [Random Number Generation](#)
- [TR1 Mathematical Special Functions](#)

#### 3.11.1 Detailed Description

Components for performing numeric operations. Includes support for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and mathematical special functions.

### 3.11.2 Bit manipulation

Collaboration diagram for Bit manipulation:



Utilities for examining and manipulating individual bits.

### 3.11.3 Complex Numbers

Collaboration diagram for Complex Numbers:



#### Classes

- class `std::complex< _Tp >`
- class `std::complex< double >`
- class `std::complex< float >`
- class `std::complex< long double >`

#### Functions

- constexpr `std::complex< double >::complex` (const `complex< long double > &`)
- constexpr `std::complex< float >::complex` (const `complex< double > &`)
- constexpr `std::complex< float >::complex` (const `complex< long double > &`)
- template<typename `_Tp`>  
`_Tp std::__complex_abs` (const `complex< _Tp > &__z`)
- template<typename `_Tp`>  
`_Tp std::__complex_arg` (const `complex< _Tp > &__z`)
- template<typename `_Tp`>  
`complex< _Tp > std::__complex_cos` (const `complex< _Tp > &__z`)
- template<typename `_Tp`>  
`complex< _Tp > std::__complex_cosh` (const `complex< _Tp > &__z`)
- template<typename `_Tp`>  
`complex< _Tp > std::__complex_exp` (const `complex< _Tp > &__z`)

- `template<typename _Tp>`  
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::tr1::conj (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Tp constexpr std::norm (const complex< _Tp > &)`
- `constexpr complex< _Tp > & std::complex< _Tp >::operator*= (const _Tp &)`
- `template<typename _Up>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator*= (const complex< _Up > &)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Up>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator+= (const complex< _Up > &)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Up>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator-= (const complex< _Up > &)`

- constexpr `complex<_Tp> & std::complex<_Tp>::operator/=(const _Tp &)`
- template<typename \_Up>  
constexpr `complex<_Tp> & std::complex<_Tp>::operator/=(const complex<_Up> &)`
- template<typename \_Tp, typename \_CharT, class \_Traits>  
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> & __os, const complex<_Tp> & __x)`
- template `ostream & std::operator<< (ostream &, const complex<double> &)`
- template `ostream & std::operator<< (ostream &, const complex<float> &)`
- template `ostream & std::operator<< (ostream &, const complex<long double> &)`
- template `wostream & std::operator<< (wostream &, const complex<double> &)`
- template `wostream & std::operator<< (wostream &, const complex<float> &)`
- template `wostream & std::operator<< (wostream &, const complex<long double> &)`
- constexpr `complex<_Tp> & std::complex<_Tp>::operator/=(const _Tp &)`
- template<typename \_Up>  
constexpr `complex<_Tp> & std::complex<_Tp>::operator/=(const complex<_Up> &)`
- template<typename \_Tp, typename \_CharT, class \_Traits>  
`basic_istream<_CharT, _Traits> & std::operator>> (basic_istream<_CharT, _Traits> & __is, complex<_Tp> & __x)`
- template `istream & std::operator>> (istream &, complex<double> &)`
- template `istream & std::operator>> (istream &, complex<float> &)`
- template `istream & std::operator>> (istream &, complex<long double> &)`
- template `wistream & std::operator>> (wistream &, complex<double> &)`
- template `wistream & std::operator>> (wistream &, complex<float> &)`
- template `wistream & std::operator>> (wistream &, complex<long double> &)`
- template<typename \_Tp>  
`complex<_Tp> std::polar(const _Tp &, const _Tp & __rho)`
- template<typename \_Tp, typename \_Up>  
`std::complex<typename __gnu_cxx::__promote_2<_Tp, _Up>::__type> std::tr1::polar(const _Tp & __rho, const _Up & __theta)`
- template<typename \_Tp>  
`complex<_Tp> std::pow(const _Tp &, const complex<_Tp> &)`
- template<typename \_Tp>  
`complex<_Tp> std::pow(const complex<_Tp> &, const _Tp &)`
- template<typename \_Tp>  
`complex<_Tp> std::pow(const complex<_Tp> &, const complex<_Tp> &)`
- template<typename \_Tp>  
`complex<_Tp> std::pow(const complex<_Tp> &, int)`
- template<typename \_Tp>  
`std::complex<_Tp> std::tr1::pow(const _Tp & __x, const std::complex<_Tp> & __y)`
- template<typename \_Tp, typename \_Up>  
`std::complex<typename __gnu_cxx::__promote_2<_Tp, _Up>::__type> std::tr1::pow(const _Tp & __x, const std::complex<_Up> & __y)`
- template<typename \_Tp>  
`std::complex<_Tp> std::tr1::pow(const std::complex<_Tp> & __x, const _Tp & __y)`
- template<typename \_Tp, typename \_Up>  
`std::complex<typename __gnu_cxx::__promote_2<_Tp, _Up>::__type> std::tr1::pow(const std::complex<_Tp> & __x, const _Up & __y)`
- template<typename \_Tp>  
`std::complex<_Tp> std::tr1::pow(const std::complex<_Tp> & __x, const std::complex<_Tp> & __y)`
- template<typename \_Tp, typename \_Up>  
`std::complex<typename __gnu_cxx::__promote_2<_Tp, _Up>::__type> std::tr1::pow(const std::complex<_Tp> & __x, const std::complex<_Up> & __y)`

- `template<typename _Tp>`  
`constexpr _Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`

### 3.11.3.1 Detailed Description

Classes and functions for complex numbers.

### 3.11.3.2 Function Documentation

#### abs()

```
template<typename _Tp>
_Tp std::abs (
    const complex< _Tp > & __z) [inline]
```

Return magnitude of z.

Referenced by `frexpl()`, `std::poisson_distribution< _IntType >::operator()()`, and `std::shuffle_order_engine< minstd_rand0, 256 >::operator()`.

#### arg()

```
template<typename _Tp>
_Tp std::arg (
    const complex< _Tp > & __z) [inline]
```

Return phase angle of z.

#### conj()

```
template<typename _Tp>
complex< _Tp > std::conj (
    const complex< _Tp > & __z) [inline], [constexpr]
```

Return complex conjugate of z.

#### cos()

```
template<typename _Tp>
complex< _Tp > std::cos (
    const complex< _Tp > & __z) [inline]
```

Return complex cosine of z.

#### cosh()

```
template<typename _Tp>
complex< _Tp > std::cosh (
    const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic cosine of z.

#### exp()

```
template<typename _Tp>
complex< _Tp > std::exp (
    const complex< _Tp > & __z) [inline]
```

Return complex base e exponential of z.

#### fabs()

```
template<typename _Tp>
std::complex< _Tp > std::tr1::fabs (
    const std::complex< _Tp > & __z) [inline]
```

`fabs(__z)` [8.1.8].

#### log()

```
template<typename _Tp>
complex< _Tp > std::log (
```



```
const complex< _Tp > & __z) [inline]
```

Return complex natural logarithm of z.

Referenced by [generate\\_canonical\(\)](#), [std::gamma\\_distribution< \\_RealType >::operator\(\)\(\)](#), [std::normal\\_distribution< \\_RealType >::operator\(\)\(\)](#), [std::poisson\\_distribution< \\_IntType >::operator\(\)\(\)](#), [std::shuffle\\_order\\_engine< minstd\\_rand0, 256 >::operator\(\)\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), and [operator<<\(\)](#).

### log10()

```
template<typename _Tp>
complex< _Tp > std::log10 (
    const complex< _Tp > & __z) [inline]
```

Return complex base 10 logarithm of z.

### norm()

```
template<typename _Tp>
_Tp constexpr std::norm (
    const complex< _Tp > & __z) [inline], [constexpr]
```

Return z magnitude squared.

### operator\*() [1/3]

```
template<typename _Tp>
complex< _Tp > std::operator* (
    const _Tp & __x,
    const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x times y.

### operator\*() [2/3]

```
template<typename _Tp>
complex< _Tp > std::operator* (
    const complex< _Tp > & __x,
    const _Tp & __y) [inline], [constexpr]
```

Return new complex value x times y.

### operator\*() [3/3]

```
template<typename _Tp>
complex< _Tp > std::operator* (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x times y.

### operator\*=() [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator*= (
    const _Tp & __t) [constexpr]
```

Multiply this complex number by a scalar.

### operator\*=() [2/2]

```
template<typename _Tp>
template<typename _Up>
```

```
complex< _Tp > & std::complex< _Tp >::operator*= (
    const complex< _Up > & __z) [constexpr]
```

Multiply this complex number by another.

#### **operator+() [1/4]**

```
template<typename _Tp>
complex< _Tp > std::operator+ (
    const _Tp & __x,
    const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x plus y.

#### **operator+() [2/4]**

```
template<typename _Tp>
complex< _Tp > std::operator+ (
    const complex< _Tp > & __x) [inline], [constexpr]
```

Return x.

#### **operator+() [3/4]**

```
template<typename _Tp>
complex< _Tp > std::operator+ (
    const complex< _Tp > & __x,
    const _Tp & __y) [inline], [constexpr]
```

Return new complex value x plus y.

#### **operator+() [4/4]**

```
template<typename _Tp>
complex< _Tp > std::operator+ (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x plus y.

#### **operator+=()**

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator+= (
    const complex< _Up > & __z) [constexpr]
```

Add another complex number to this one.

#### **operator-() [1/4]**

```
template<typename _Tp>
complex< _Tp > std::operator- (
    const _Tp & __x,
    const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x minus y.

#### **operator-() [2/4]**

```
template<typename _Tp>
complex< _Tp > std::operator- (
    const complex< _Tp > & __x) [inline], [constexpr]
```

Return complex negation of x.

#### **operator-() [3/4]**

```
template<typename _Tp>
complex< _Tp > std::operator- (
    const complex< _Tp > & __x,
    const _Tp & __y) [inline], [constexpr]
```

Return new complex value x minus y.

#### **operator-() [4/4]**

```
template<typename _Tp>
complex< _Tp > std::operator- (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x minus y.

#### **operator-=( )**

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator-= (
    const complex< _Up > & __z) [constexpr]
```

Subtract another complex number from this one.

#### **operator/( ) [1/3]**

```
template<typename _Tp>
complex< _Tp > std::operator/ (
    const _Tp & __x,
    const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x divided by y.

#### **operator/( ) [2/3]**

```
template<typename _Tp>
complex< _Tp > std::operator/ (
    const complex< _Tp > & __x,
    const _Tp & __y) [inline], [constexpr]
```

Return new complex value x divided by y.

#### **operator/( ) [3/3]**

```
template<typename _Tp>
complex< _Tp > std::operator/ (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x divided by y.

#### **operator/=( ) [1/2]**

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator/= (
    const _Tp & __t) [constexpr]
```

Divide this complex number by a scalar.

**operator/=(** [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator/= (
    const complex< _Up > & __z) [constexpr]
```

Divide this complex number by another.

**operator<<()**

```
template<typename _Tp, typename _CharT, class _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const complex< _Tp > & __x)
```

Insertion operator for complex values.

**operator=(** [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator= (
    const _Tp & __t) [constexpr]
```

Assign a scalar to this complex number.

**operator=(** [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator= (
    const complex< _Up > & __z) [constexpr]
```

Assign another complex number to this one.

**operator==(** [1/2]

```
template<typename _Tp>
bool std::operator== (
    const complex< _Tp > & __x,
    const _Tp & __y) [inline], [constexpr]
```

Return true if x is equal to y.

**operator==(** [2/2]

```
template<typename _Tp>
bool std::operator== (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y) [inline], [constexpr]
```

Return true if x is equal to y.

**operator>>()**

```
template<typename _Tp, typename _CharT, class _Traits>
basic_istream< _CharT, _Traits > & std::operator>> (
    basic_istream< _CharT, _Traits > & __is,
    complex< _Tp > & __x)
```

Extraction operator for complex values.

**polar()**

```
template<typename _Tp>
complex< _Tp > std::polar (
    const _Tp & __rho,
    const _Tp & __theta = _Tp(0)) [inline]
```

Return complex with magnitude *rho* and angle *theta*.

**pow() [1/5]**

```
template<typename _Tp>
complex< _Tp > std::pow (
    const _Tp & __x,
    const complex< _Tp > & __y) [inline]
```

Return *x* to the *y*<sup>th</sup> power.

**pow() [2/5]**

```
template<typename _Tp>
complex< _Tp > std::pow (
    const complex< _Tp > & __x,
    const _Tp & __y)
```

Return *x* to the *y*<sup>th</sup> power.

**pow() [3/5]**

```
template<typename _Tp>
complex< _Tp > std::pow (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y) [inline]
```

Return *x* to the *y*<sup>th</sup> power.

**pow() [4/5]**

```
template<typename _Tp>
complex< _Tp > std::pow (
    const complex< _Tp > & __z,
    int __n) [inline]
```

Return *x* to the *y*<sup>th</sup> power.

Referenced by [std::gamma\\_distribution< \\_RealType >::operator\(\)\(\)](#), and [operator<<\(\)](#).

**pow() [5/5]**

```
template<typename _Tp, typename _Up>
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (
    const std::complex< _Tp > & __x,
    const _Up & __y) [inline]
```

Additional overloads [8.1.9].

**sin()**

```
template<typename _Tp>
complex< _Tp > std::sin (
    const complex< _Tp > & __z) [inline]
```

Return complex sine of *z*.

**sinh()**

```
template<typename _Tp>
complex< _Tp > std::sinh (
    const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic sine of z.

**sqrt()**

```
template<typename _Tp>
complex< _Tp > std::sqrt (
    const complex< _Tp > & __z) [inline]
```

Return complex square root of z.

Referenced by `std::normal_distribution< _RealType >::operator()()`, and `std::student_t_distribution< _RealType >::operator()()`.

**tan()**

```
template<typename _Tp>
complex< _Tp > std::tan (
    const complex< _Tp > & __z) [inline]
```

Return complex tangent of z.

Referenced by `operator<<()`.

**tanh()**

```
template<typename _Tp>
complex< _Tp > std::tanh (
    const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic tangent of z.

**3.11.4 Decimal Floating-Point Arithmetic**

Collaboration diagram for Decimal Floating-Point Arithmetic:

**Namespaces**

- namespace `std::decimal`

**3.11.4.1 Detailed Description**

Classes and functions for decimal floating-point arithmetic.

### 3.11.5 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



#### Functions

- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)`
- `float __gnu_cxx::airy_aif (float __x)`
- `long double __gnu_cxx::airy_ail (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)`
- `float __gnu_cxx::airy_bif (float __x)`
- `long double __gnu_cxx::airy_bil (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tpa, typename _Tpb>`  
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)`
- `float std::betaf (float __a, float __b)`
- `long double std::betal (long double __a, long double __b)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)`
- `float std::comp_ellint_1f (float __k)`
- `long double std::comp_ellint_1l (long double __k)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)`
- `float std::comp_ellint_2f (float __k)`
- `long double std::comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn>`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (_Tp __k, _Tpn __nu)`
- `float std::comp_ellint_3f (float __k, float __nu)`
- `long double std::comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)`

- long double `__gnu_cxx::conf_hypergl` (long double \_\_a, long double \_\_c, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_bessel_i` (\_Tpnu \_\_nu, \_Tp \_\_x)
- float `std::cyl_bessel_if` (float \_\_nu, float \_\_x)
- long double `std::cyl_bessel_il` (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_bessel_j` (\_Tpnu \_\_nu, \_Tp \_\_x)
- float `std::cyl_bessel_jf` (float \_\_nu, float \_\_x)
- long double `std::cyl_bessel_jl` (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_bessel_k` (\_Tpnu \_\_nu, \_Tp \_\_x)
- float `std::cyl_bessel_kf` (float \_\_nu, float \_\_x)
- long double `std::cyl_bessel_kl` (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_neumann` (\_Tpnu \_\_nu, \_Tp \_\_x)
- float `std::cyl_neumannf` (float \_\_nu, float \_\_x)
- long double `std::cyl_neumannl` (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp>  
`__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::ellint_1` (\_Tp \_\_k, \_Tpp \_\_phi)
- float `std::ellint_1f` (float \_\_k, float \_\_phi)
- long double `std::ellint_1l` (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp>  
`__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::ellint_2` (\_Tp \_\_k, \_Tpp \_\_phi)
- float `std::ellint_2f` (float \_\_k, float \_\_phi)
- long double `std::ellint_2l` (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp>  
`__gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type std::ellint_3` (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float `std::ellint_3f` (float \_\_k, float \_\_nu, float \_\_phi)
- long double `std::ellint_3l` (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp>  
`__gnu_cxx::__promote<_Tp>::__type std::expint` (\_Tp \_\_x)
- float `std::expintf` (float \_\_x)
- long double `std::expintl` (long double \_\_x)
- template<typename \_Tp>  
`__gnu_cxx::__promote<_Tp>::__type std::hermite` (unsigned int \_\_n, \_Tp \_\_x)
- float `std::hermitef` (unsigned int \_\_n, float \_\_x)
- long double `std::hermitel` (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp>  
`__gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc, _Tp>::__type __gnu_cxx::hyperg` (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float `__gnu_cxx::hypergf` (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double `__gnu_cxx::hypergl` (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp>  
`__gnu_cxx::__promote<_Tp>::__type std::laguerre` (unsigned int \_\_n, \_Tp \_\_x)
- float `std::laguerref` (unsigned int \_\_n, float \_\_x)
- long double `std::laguerrel` (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
`__gnu_cxx::__promote<_Tp>::__type std::legendre` (unsigned int \_\_l, \_Tp \_\_x)
- float `std::legendref` (unsigned int \_\_l, float \_\_x)
- long double `std::legendrel` (unsigned int \_\_l, long double \_\_x)
- template<typename \_Tp>  
`__gnu_cxx::__promote<_Tp>::__type std::riemann_zeta` (\_Tp \_\_s)



- float [std::riemann\\_zetaf](#) (float \_\_s)
- long double [std::riemann\\_zetal](#) (long double \_\_s)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_bessel](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::sph\\_besself](#) (unsigned int \_\_n, float \_\_x)
- long double [std::sph\\_bessell](#) (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_legendre](#) (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float [std::sph\\_legendref](#) (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double [std::sph\\_legendrel](#) (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_neumann](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::sph\\_neumannf](#) (unsigned int \_\_n, float \_\_x)
- long double [std::sph\\_neumannl](#) (unsigned int \_\_n, long double \_\_x)

### 3.11.5.1 Detailed Description

### 3.11.5.2 Mathematical Special Functions

A collection of advanced mathematical special functions, defined by ISO/IEC IS 29124 and then added to ISO C++ 2017.

#### 3.11.5.2.1 Introduction and History

The first significant library upgrade on the road to C++2011, [TR1](#), included a set of 23 mathematical functions that significantly extended the standard transcendental functions inherited from C and declared in `<cmath>`.

Although most components from TR1 were eventually adopted for C++11 these math functions were left behind out of concern for implementability. The math functions were published as a separate international standard [IS 29124 - Extensions to the C++ Library to Support Mathematical Special Functions](#).

For C++17 these functions were incorporated into the main standard.

#### 3.11.5.2.2 Contents

The following functions are implemented in namespace `std`:

- `assoc_laguerre` - Associated Laguerre functions
- `assoc_legendre` - Associated Legendre functions
- `beta` - Beta functions
- `comp_ellint_1` - Complete elliptic functions of the first kind
- `comp_ellint_2` - Complete elliptic functions of the second kind
- `comp_ellint_3` - Complete elliptic functions of the third kind
- `cyl_bessel_i` - Regular modified cylindrical Bessel functions
- `cyl_bessel_j` - Cylindrical Bessel functions of the first kind
- `cyl_bessel_k` - Irregular modified cylindrical Bessel functions
- `cyl_neumann` - Cylindrical Neumann functions or Cylindrical Bessel functions of the second kind
- `ellint_1` - Incomplete elliptic functions of the first kind
- `ellint_2` - Incomplete elliptic functions of the second kind

- `ellint_3` - Incomplete elliptic functions of the third kind
- `expint` - The exponential integral
- `hermite` - Hermite polynomials
- `laguerre` - Laguerre functions
- `legendre` - Legendre polynomials
- `riemann_zeta` - The Riemann zeta function
- `sph_bessel` - Spherical Bessel functions
- `sph_legendre` - Spherical Legendre functions
- `sph_neumann` - Spherical Neumann functions

The hypergeometric functions were stricken from the TR29124 and C++17 versions of this math library because of implementation concerns. However, since they were in the TR1 version and since they are popular we kept them as an extension in namespace `__gnu_cxx`:

- [conf\\_hyperg](#) - Confluent hypergeometric functions
- [hyperg](#) - Hypergeometric functions

#### 3.11.5.2.3 Argument Promotion

The arguments supplied to the non-suffixed functions will be promoted according to the following rules:

1. If any argument intended to be floating point is given an integral value That integral value is promoted to double.
2. All floating point arguments are promoted up to the largest floating point precision among them.

#### 3.11.5.2.4 NaN Arguments

If any of the floating point arguments supplied to these functions is invalid or NaN (`std::numeric_limits<Tp>::quiet_NaN`), the value NaN is returned.

#### 3.11.5.2.5 Implementation

We strive to implement the underlying math with type generic algorithms to the greatest extent possible. In practice, the functions are thin wrappers that dispatch to function templates. Type dependence is controlled with `std::numeric_limits` and functions thereof.

We don't promote `float` to `double` or `double` to `long double` reflexively. The goal is for `float` functions to operate more quickly, at the cost of `float` accuracy and possibly a smaller domain of validity. Similarly, `long double` should give you more dynamic range and slightly more precision than `double` on many systems.

#### 3.11.5.2.6 Testing

These functions have been tested against equivalent implementations from the [Gnu Scientific Library](#), [GSL](#) and [Boost](#) and the ratio

$$\frac{|f - f_{test}|}{|f_{test}|}$$

is generally found to be within  $10^{-15}$  for 64-bit double on linux-x86\_64 systems over most of the ranges of validity.

**Todo** Provide accuracy comparisons on a per-function basis for a small number of targets.

### 3.11.5.2.7 General Bibliography

See also

Abramowitz and Stegun: Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables  
Edited by Milton Abramowitz and Irene A. Stegun, National Bureau of Standards Applied Mathematics Series - 55  
Issued June 1964, Tenth Printing, December 1972, with corrections Electronic versions of A&S abound including both pdf and navigable html.

for example <http://people.math.sfu.ca/~cbm/aands/>

The old A&S has been redone as the NIST Digital Library of Mathematical Functions: <http://dlmf.nist.gov/> This version is far more navigable and includes more recent work.

An Atlas of Functions: with Equator, the Atlas Function Calculator 2nd Edition, by Oldham, Keith B., Myland, Jan, Spanier, Jerome

Asymptotics and Special Functions by Frank W. J. Olver, Academic Press, 1974

Numerical Recipes in C, The Art of Scientific Computing, by William H. Press, Second Ed., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Cambridge University Press, 1992

The Special Functions and Their Approximations: Volumes 1 and 2, by Yudell L. Luke, Academic Press, 1969

### 3.11.5.3 Function Documentation

#### airy\_ai()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (
    _Tp __x) [inline]
```

Return the Airy function  $Ai(x)$  of real argument x.

#### airy\_aif()

```
float __gnu_cxx::airy_aif (
    float __x) [inline]
```

Return the Airy function  $Ai(x)$  of float argument x.

#### airy\_ail()

```
long double __gnu_cxx::airy_ail (
    long double __x) [inline]
```

Return the Airy function  $Ai(x)$  of long double argument x.

#### airy\_bi()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (
    _Tp __x) [inline]
```

Return the Airy function  $Bi(x)$  of real argument x.

#### airy\_bif()

```
float __gnu_cxx::airy_bif (
    float __x) [inline]
```

Return the Airy function  $Bi(x)$  of float argument x.

**airy\_bil()**

```
long double __gnu_cxx::airy_bil (
    long double __x) [inline]
```

Return the Airy function  $Bi(x)$  of long double argument  $x$ .

**assoc\_laguerre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (
    unsigned int __n,
    unsigned int __m,
    _Tp __x) [inline]
```

Return the associated Laguerre polynomial of nonnegative order  $n$ , nonnegative degree  $m$  and real argument  $x \leftarrow : L_n^m(x)$ .

The associated Laguerre function of real degree  $\alpha$ ,  $L_n^\alpha(x)$ , is defined by

$$L_n^\alpha(x) = \frac{(\alpha+1)_n}{n!} {}_1F_1(-n; \alpha+1; x)$$

where  $(\alpha)_n$  is the Pochhammer symbol and  ${}_1F_1(a; c; x)$  is the confluent hypergeometric function.

The associated Laguerre polynomial is defined for integral degree  $\alpha = m$  by:

$$L_n^m(x) = (-1)^m \frac{d^m}{dx^m} L_{n+m}(x)$$

where the Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x})$$

and  $x \geq 0$ .

See also

laguerre for details of the Laguerre function of degree  $n$

**Template Parameters**

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

**Parameters**

<code>__n</code>	The order of the Laguerre function, <code>__n</code> $\geq 0$ .
<code>__m</code>	The degree of the Laguerre function, <code>__m</code> $\geq 0$ .
<code>__x</code>	The argument of the Laguerre function, <code>__x</code> $\geq 0$ .

**Exceptions**

<code>std::domain_error</code>	if <code>__x</code> $< 0$ .
--------------------------------	-----------------------------

**assoc\_laguerref()**

```
float std::assoc_laguerref (
    unsigned int __n,
    unsigned int __m,
    float __x) [inline]
```

Return the associated Laguerre polynomial of order  $n$ , degree  $m$ :  $L_n^m(x)$  for `float` argument.

**See also**

`assoc_laguerre` for more details.

**assoc\_laguerrel()**

```
long double std::assoc_laguerrel (
    unsigned int __n,
    unsigned int __m,
    long double __x) [inline]
```

Return the associated Laguerre polynomial of order  $n$ , degree  $m$ :  $L_n^m(x)$ .

**See also**

`assoc_laguerre` for more details.

**assoc\_legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __x) [inline]
```

Return the associated Legendre function of degree  $l$  and order  $m$ .

The associated Legendre function is derived from the Legendre function  $P_l(x)$  by the Rodrigues formula:

$$P_l^m(x) = (1 - x^2)^{m/2} \frac{d^m}{dx^m} P_l(x)$$

**See also**

`legendre` for details of the Legendre function of degree  $l$

**Template Parameters**

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

**Parameters**

<code>__l</code>	The degree <code>__l</code> $\geq 0$ .
<code>__m</code>	The order <code>__m</code> $\leq l$ .
<code>__x</code>	The argument, <code>abs (__x)</code> $\leq 1$ .

## Exceptions

<code>std::domain_error</code>	if <code>abs(__x) &gt; 1.</code>
--------------------------------	----------------------------------

**assoc\_legendref()**

```
float std::assoc_legendref (
    unsigned int __l,
    unsigned int __m,
    float __x) [inline]
```

Return the associated Legendre function of degree `l` and order `m` for `float` argument.

## See also

`assoc_legendre` for more details.

**assoc\_legendrel()**

```
long double std::assoc_legendrel (
    unsigned int __l,
    unsigned int __m,
    long double __x) [inline]
```

Return the associated Legendre function of degree `l` and order `m`.

## See also

`assoc_legendre` for more details.

**beta()**

```
template<typename _Tpa, typename _Tpb>
__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (
    _Tpa __a,
    _Tpb __b) [inline]
```

Return the beta function,  $B(a, b)$ , for real parameters `a`, `b`.

The beta function is defined by

$$B(a, b) = \int_0^1 t^{a-1} (1-t)^{b-1} dt = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

where  $a > 0$  and  $b > 0$

## Template Parameters

<code>_Tpa</code>	The floating-point type of the parameter <code>__a</code> .
<code>_Tpb</code>	The floating-point type of the parameter <code>__b</code> .

## Parameters

<code>__a</code>	The first argument of the beta function, <code>__a &gt; 0</code> .
<code>__b</code>	The second argument of the beta function, <code>__b &gt; 0</code> .

**Exceptions**

<code>std::domain_error</code>	if <code>__a &lt; 0</code> or <code>__b &lt; 0</code> .
--------------------------------	---

**betaf()**

```
float std::betaf (
    float __a,
    float __b) [inline]
```

Return the beta function,  $B(a, b)$ , for `float` parameters `a`, `b`.

**See also**

`beta` for more details.

**betal()**

```
long double std::betal (
    long double __a,
    long double __b) [inline]
```

Return the beta function,  $B(a, b)$ , for long double parameters `a`, `b`.

**See also**

`beta` for more details.

**comp\_ellint\_1()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (
    _Tp __k) [inline]
```

Return the complete elliptic integral of the first kind  $K(k)$  for real modulus `k`.

The complete elliptic integral of the first kind is defined as

$$K(k) = F(k, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$$

where  $F(k, \phi)$  is the incomplete elliptic integral of the first kind and the modulus  $|k| \leq 1$ .

**See also**

`ellint_1` for details of the incomplete elliptic function of the first kind.

**Template Parameters**

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
------------------	---

**Parameters**

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
------------------	---

## Exceptions

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1</code> .
--------------------------------	------------------------------------

**comp\_ellint\_1f()**

```
float std::comp_ellint_1f (
    float __k) [inline]
```

Return the complete elliptic integral of the first kind  $E(k)$  for `float` modulus `k`.

## See also

`comp_ellint_1` for details.

**comp\_ellint\_1l()**

```
long double std::comp_ellint_1l (
    long double __k) [inline]
```

Return the complete elliptic integral of the first kind  $E(k)$  for `long double` modulus `k`.

## See also

`comp_ellint_1` for details.

**comp\_ellint\_2()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (
    _Tp __k) [inline]
```

Return the complete elliptic integral of the second kind  $E(k)$  for real modulus `k`.  
The complete elliptic integral of the second kind is defined as

$$E(k) = E(k, \pi/2) = \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2 \theta} d\theta$$

where  $E(k, \phi)$  is the incomplete elliptic integral of the second kind and the modulus  $|k| \leq 1$ .

## See also

`ellint_2` for details of the incomplete elliptic function of the second kind.

## Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
------------------	---

## Parameters

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
------------------	---



**Exceptions**

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1</code> .
--------------------------------	------------------------------------

**comp\_ellint\_2f()**

```
float std::comp_ellint_2f (
    float __k) [inline]
```

Return the complete elliptic integral of the second kind  $E(k)$  for `float` modulus `k`.

**See also**

`comp_ellint_2` for details.

**comp\_ellint\_2l()**

```
long double std::comp_ellint_2l (
    long double __k) [inline]
```

Return the complete elliptic integral of the second kind  $E(k)$  for long double modulus `k`.

**See also**

`comp_ellint_2` for details.

**comp\_ellint\_3()**

```
template<typename _Tp, typename _Tpn>
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (
    _Tp __k,
    _Tpn __nu) [inline]
```

Return the complete elliptic integral of the third kind  $\Pi(k, \nu) = \Pi(k, \nu, \pi/2)$  for real modulus `k`.

The complete elliptic integral of the third kind is defined as

$$\Pi(k, \nu) = \Pi(k, \nu, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}$$

where  $\Pi(k, \nu, \phi)$  is the incomplete elliptic integral of the second kind and the modulus  $|k| \leq 1$ .

**See also**

`ellint_3` for details of the incomplete elliptic function of the third kind.

**Template Parameters**

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>_Tpn</code>	The floating-point type of the argument <code>__nu</code> .

**Parameters**

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
<code>__nu</code>	The argument

## Exceptions

<code>std::domain_error</code>	if <code>abs(__k) &gt; 1</code> .
--------------------------------	-----------------------------------

**comp\_ellint\_3f()**

```
float std::comp_ellint_3f (
    float __k,
    float __nu) [inline]
```

Return the complete elliptic integral of the third kind  $\Pi(k, \nu)$  for `float` modulus `k`.

## See also

`comp_ellint_3` for details.

**comp\_ellint\_3l()**

```
long double std::comp_ellint_3l (
    long double __k,
    long double __nu) [inline]
```

Return the complete elliptic integral of the third kind  $\Pi(k, \nu)$  for `long double` modulus `k`.

## See also

`comp_ellint_3` for details.

**conf\_hyperg()**

```
template<typename _Tpa, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (
    _Tpa __a,
    _Tpc __c,
    _Tp __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of real numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

The confluent hypergeometric function is defined by

$${}_1F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$

## Parameters

<code>↔ _a</code>	The numeratorial parameter
<code>↔ _c</code>	The denominatorial parameter
<code>↔ _x</code>	The argument

**conf\_hypergf()**

```
float __gnu_cxx::conf_hypergf (
    float __a,
    float __c,
    float __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of `float` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

**See also**

`conf_hyperg` for details.

**conf\_hypergl()**

```
long double __gnu_cxx::conf_hypergl (
    long double __a,
    long double __c,
    long double __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of `long double` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

**See also**

`conf_hyperg` for details.

**cyl\_bessel\_i()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (
    _Tpnu __nu,
    _Tp __x) [inline]
```

Return the regular modified Bessel function  $I_\nu(x)$  for real order  $\nu$  and argument  $x \geq 0$ . The regular modified cylindrical Bessel function is:

$$I_\nu(x) = i^{-\nu} J_\nu(ix) = \sum_{k=0}^{\infty} \frac{(x/2)^{\nu+2k}}{k! \Gamma(\nu + k + 1)}$$

**Template Parameters**

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

**Parameters**

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x</code> $\geq 0$

**Exceptions**

<code>std::domain_error</code>	if <code>__x</code> $< 0$ .
--------------------------------	-----------------------------

**cyl\_bessel\_if()**

```
float std::cyl_bessel_if (
    float __nu,
    float __x) [inline]
```

Return the regular modified Bessel function  $I_\nu(x)$  for float order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_i` for setails.

**cyl\_bessel\_il()**

```
long double std::cyl_bessel_il (
    long double __nu,
    long double __x) [inline]
```

Return the regular modified Bessel function  $I_\nu(x)$  for long double order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_i` for setails.

**cyl\_bessel\_j()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (
    _Tpnu __nu,
    _Tp __x) [inline]
```

Return the Bessel function  $J_\nu(x)$  of real order  $\nu$  and argument  $x \geq 0$ .

The cylindrical Bessel function is:

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{\nu+2k}}{k! \Gamma(\nu + k + 1)}$$

**Template Parameters**

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

**Parameters**

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x</code> $\geq 0$

**Exceptions**

<code>std::domain_error</code>	if <code>__x</code> $< 0$ .
--------------------------------	-----------------------------

**cyl\_bessel\_jf()**

```
float std::cyl_bessel_jf (
    float __nu,
    float __x) [inline]
```

Return the Bessel function of the first kind  $J_\nu(x)$  for float order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_j` for setails.

**cyl\_bessel\_jl()**

```
long double std::cyl_bessel_jl (
    long double __nu,
    long double __x) [inline]
```

Return the Bessel function of the first kind  $J_\nu(x)$  for long double order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_j` for setails.

**cyl\_bessel\_k()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k (
    _Tpnu __nu,
    _Tp __x) [inline]
```

Return the irregular modified Bessel function  $K_\nu(x)$  of real order  $\nu$  and argument  $x$ .

The irregular modified Bessel function is defined by:

$$K_\nu(x) = \frac{\pi}{2} \frac{I_{-\nu}(x) - I_\nu(x)}{\sin \nu\pi}$$

where for integral  $\nu = n$  a limit is taken:  $\lim_{\nu \rightarrow n}$ . For negative argument we have simply:

$$K_{-\nu}(x) = K_\nu(x)$$

**Template Parameters**

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

**Parameters**

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x</code> $\geq 0$

**Exceptions**

<code>std::domain_error</code>	if <code>__x</code> $< 0$ .
--------------------------------	-----------------------------

**cyl\_bessel\_kf()**

```
float std::cyl_bessel_kf (
    float __nu,
    float __x) [inline]
```

Return the irregular modified Bessel function  $K_\nu(x)$  for float order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_k` for setails.

**cyl\_bessel\_kl()**

```
long double std::cyl_bessel_kl (
    long double __nu,
    long double __x) [inline]
```

Return the irregular modified Bessel function  $K_\nu(x)$  for long double order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_k` for setails.

**cyl\_neumann()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann (
    _Tpnu __nu,
    _Tp __x) [inline]
```

Return the Neumann function  $N_\nu(x)$  of real order  $\nu$  and argument  $x \geq 0$ .

The Neumann function is defined by:

$$N_\nu(x) = \frac{J_\nu(x) \cos \nu\pi - J_{-\nu}(x)}{\sin \nu\pi}$$

where  $x \geq 0$  and for integral order  $\nu = n$  a limit is taken:  $\lim_{\nu \rightarrow n}$ .

**Template Parameters**

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

**Parameters**

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x</code> $\geq 0$

**Exceptions**

<code>std::domain_error</code>	if <code>__x</code> $< 0$ .
--------------------------------	-----------------------------

**cyl\_neumannf()**

```
float std::cyl_neumannf (
    float __nu,
    float __x) [inline]
```

Return the Neumann function  $N_\nu(x)$  of `float` order  $\nu$  and argument  $x$ .

See also

`cyl_neumann` for setails.

**cyl\_neumannl()**

```
long double std::cyl_neumannl (
    long double __nu,
    long double __x) [inline]
```

Return the Neumann function  $N_\nu(x)$  of `long double` order  $\nu$  and argument  $x$ .

See also

`cyl_neumann` for setails.

**ellint\_1()**

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 (
    _Tp __k,
    _Tpp __phi) [inline]
```

Return the incomplete elliptic integral of the first kind  $F(k, \phi)$  for `real` modulus  $k$  and angle  $\phi$ .  
The incomplete elliptic integral of the first kind is defined as

$$F(k, \phi) = \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$$

For  $\phi = \pi/2$  this becomes the complete elliptic integral of the first kind,  $K(k)$ .

See also

`comp_ellint_1`.

**Template Parameters**

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>_Tpp</code>	The floating-point type of the angle <code>__phi</code> .

**Parameters**

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
<code>__phi</code>	The integral limit argument in radians

**Exceptions**

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1</code> .
--------------------------------	------------------------------------

**ellint\_1f()**

```
float std::ellint_1f (
    float __k,
    float __phi) [inline]
```

Return the incomplete elliptic integral of the first kind  $E(k, \phi)$  for `float` modulus  $k$  and angle  $\phi$ .

See also

`ellint_1` for details.

**ellint\_1l()**

```
long double std::ellint_1l (
    long double __k,
    long double __phi) [inline]
```

Return the incomplete elliptic integral of the first kind  $E(k, \phi)$  for `long double` modulus  $k$  and angle  $\phi$ .

See also

`ellint_1` for details.

**ellint\_2()**

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 (
    _Tp __k,
    _Tpp __phi) [inline]
```

Return the incomplete elliptic integral of the second kind  $E(k, \phi)$ .

The incomplete elliptic integral of the second kind is defined as

$$E(k, \phi) = \int_0^\phi \sqrt{1 - k^2 \sin^2 \theta}$$

For  $\phi = \pi/2$  this becomes the complete elliptic integral of the second kind,  $E(k)$ .

See also

`comp_ellint_2`.

**Template Parameters**

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>_Tpp</code>	The floating-point type of the angle <code>__phi</code> .

**Parameters**

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
<code>__phi</code>	The integral limit argument in radians

**Returns**

The elliptic function of the second kind.



## Exceptions

<code>std::domain_error</code>	if <code>abs(__k) &gt; 1</code> .
--------------------------------	-----------------------------------

**ellint\_2f()**

```
float std::ellint_2f (
    float __k,
    float __phi) [inline]
```

Return the incomplete elliptic integral of the second kind  $E(k, \phi)$  for `float` argument.

## See also

`ellint_2` for details.

**ellint\_2l()**

```
long double std::ellint_2l (
    long double __k,
    long double __phi) [inline]
```

Return the incomplete elliptic integral of the second kind  $E(k, \phi)$ .

## See also

`ellint_2` for details.

**ellint\_3()**

```
template<typename _Tp, typename _Tpn, typename _Tpp>
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 (
    _Tp __k,
    _Tpn __nu,
    _Tpp __phi) [inline]
```

Return the incomplete elliptic integral of the third kind  $\Pi(k, \nu, \phi)$ .

The incomplete elliptic integral of the third kind is defined by:

$$\Pi(k, \nu, \phi) = \int_0^\phi \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}$$

For  $\phi = \pi/2$  this becomes the complete elliptic integral of the third kind,  $\Pi(k, \nu)$ .

## See also

`comp_ellint_3`.

## Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>_Tpn</code>	The floating-point type of the argument <code>__nu</code> .
<code>_Tpp</code>	The floating-point type of the angle <code>__phi</code> .

## Parameters

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
<code>__nu</code>	The second argument
<code>__phi</code>	The integral limit argument in radians

## Returns

The elliptic function of the third kind.

## Exceptions

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1</code> .
--------------------------------	------------------------------------

**ellint\_3f()**

```
float std::ellint_3f (
    float __k,
    float __nu,
    float __phi) [inline]
```

Return the incomplete elliptic integral of the third kind  $\Pi(k, \nu, \phi)$  for `float` argument.

## See also

`ellint_3` for details.

**ellint\_3l()**

```
long double std::ellint_3l (
    long double __k,
    long double __nu,
    long double __phi) [inline]
```

Return the incomplete elliptic integral of the third kind  $\Pi(k, \nu, \phi)$ .

## See also

`ellint_3` for details.

**expint()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::expint (
    _Tp __x) [inline]
```

Return the exponential integral  $Ei(x)$  for real argument `x`.

The exponential integral is given by

$$Ei(x) = - \int_{-x}^{\infty} \frac{e^t}{t} dt$$

## Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

## Parameters

<code>__x</code>	The argument of the exponential integral function.
------------------	--

**expintf()**

```
float std::expintf (
    float __x) [inline]
```

Return the exponential integral  $Ei(x)$  for `float` argument `x`.

## See also

`expint` for details.

**expintl()**

```
long double std::expintl (
    long double __x) [inline]
```

Return the exponential integral  $Ei(x)$  for `long double` argument `x`.

## See also

`expint` for details.

**hermite()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::hermite (
    unsigned int __n,
    _Tp __x) [inline]
```

Return the Hermite polynomial  $H_n(x)$  of order `n` and `real` argument `x`.

The Hermite polynomial is defined by:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2}$$

The Hermite polynomial obeys a reflection formula:

$$H_n(-x) = (-1)^n H_n(x)$$

## Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

## Parameters

<code>__n</code>	The order
<code>__x</code>	The argument

**hermitef()**

```
float std::hermitef (
    unsigned int __n,
    float __x) [inline]
```

Return the Hermite polynomial  $H_n(x)$  of nonnegative order n and float argument x.

**See also**

[hermite](#) for details.

**hermitel()**

```
long double std::hermitel (
    unsigned int __n,
    long double __x) [inline]
```

Return the Hermite polynomial  $H_n(x)$  of nonnegative order n and long double argument x.

**See also**

[hermite](#) for details.

**hyperg()**

```
template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (
    _Tpa __a,
    _Tpb __b,
    _Tpc __c,
    _Tp __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of real numeratorial parameters a and b, denominatorial parameter c, and argument x.

The hypergeometric function is defined by

$${}_2F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$

**Parameters**

<a href="#"></a> <code>_a</code>	The first numeratorial parameter
<a href="#"></a> <code>_b</code>	The second numeratorial parameter
<a href="#"></a> <code>_c</code>	The denominatorial parameter
<a href="#"></a> <code>_x</code>	The argument

**hypergf()**

```
float __gnu_cxx::hypergf (
    float __a,
```

```
float __b,
float __c,
float __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of @ float numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

### hypergl()

```
long double __gnu_cxx::hypergl (
    long double __a,
    long double __b,
    long double __c,
    long double __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of long double numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

### laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::laguerre (
    unsigned int __n,
    _Tp __x) [inline]
```

Returns the Laguerre polynomial  $L_n(x)$  of nonnegative degree n and real argument  $x \geq 0$ . The Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x})$$

#### Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

#### Parameters

<code>__n</code>	The nonnegative order
<code>__x</code>	The argument <code>__x</code> $\geq 0$

#### Exceptions

<code>std::domain_error</code>	if <code>__x &lt; 0</code> .
--------------------------------	------------------------------

**laguerref()**

```
float std::laguerref (
    unsigned int __n,
    float __x) [inline]
```

Returns the Laguerre polynomial  $L_n(x)$  of nonnegative degree  $n$  and float argument  $x \geq 0$ .

See also

laguerre for more details.

**laguerrel()**

```
long double std::laguerrel (
    unsigned int __n,
    long double __x) [inline]
```

Returns the Laguerre polynomial  $L_n(x)$  of nonnegative degree  $n$  and long double argument  $x \geq 0$ .

See also

laguerre for more details.

**legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::legendre (
    unsigned int __l,
    _Tp __x) [inline]
```

Return the Legendre polynomial  $P_l(x)$  of nonnegative degree  $l$  and real argument  $|x| \leq 1$ .

The Legendre function of order  $l$  and argument  $x$ ,  $P_l(x)$ , is defined by:

$$P_l(x) = \frac{1}{2^l l!} \frac{d^l}{dx^l} (x^2 - 1)^l$$

**Template Parameters**

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

**Parameters**

<code>__l</code>	The degree $l \geq 0$
<code>__x</code>	The argument $\text{abs}(\text{__x}) \leq 1$

**Exceptions**

<code>std::domain_error</code>	if $\text{abs}(\text{__x}) > 1$
--------------------------------	---------------------------------

**legendref()**

```
float std::legendref (
    unsigned int __l,
    float __x) [inline]
```

Return the Legendre polynomial  $P_l(x)$  of nonnegative degree  $l$  and float argument  $|x| \leq 0$ .

See also

legendre for more details.

**legendrel()**

```
long double std::legendrel (
    unsigned int __l,
    long double __x) [inline]
```

Return the Legendre polynomial  $P_l(x)$  of nonnegative degree  $l$  and long double argument  $|x| \leq 0$ .

See also

legendre for more details.

**riemann\_zeta()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::riemann_zeta (
    _Tp __s) [inline]
```

Return the Riemann zeta function  $\zeta(s)$  for real argument  $s$ .

The Riemann zeta function is defined by:

$$\zeta(s) = \sum_{k=1}^{\infty} k^{-s} \text{ for } s > 1$$

and

$$\zeta(s) = \frac{1}{1-2^{1-s}} \sum_{k=1}^{\infty} (-1)^{k-1} k^{-s} \text{ for } 0 \leq s \leq 1$$

For  $s < 1$  use the reflection formula:

$$\zeta(s) = 2^s \pi^{s-1} \sin\left(\frac{\pi s}{2}\right) \Gamma(1-s) \zeta(1-s)$$

**Template Parameters**

<code>_Tp</code>	The floating-point type of the argument <code>__s</code> .
------------------	--

**Parameters**

<code>__s</code>	The argument $s \neq 1$
------------------	-------------------------

**riemann\_zetaf()**

```
float std::riemann_zetaf (
    float __s) [inline]
```

Return the Riemann zeta function  $\zeta(s)$  for float argument  $s$ .

**See also**

`riemann_zeta` for more details.

**riemann\_zetal()**

```
long double std::riemann_zetal (
    long double __s) [inline]
```

Return the Riemann zeta function  $\zeta(s)$  for long double argument  $s$ .

**See also**

`riemann_zeta` for more details.

**sph\_bessel()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_bessel (
    unsigned int __n,
    _Tp __x) [inline]
```

Return the spherical Bessel function  $j_n(x)$  of nonnegative order  $n$  and real argument  $x \geq 0$ .  
The spherical Bessel function is defined by:

$$j_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} J_{n+1/2}(x)$$

**Template Parameters**

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

**Parameters**

<code>__n</code>	The integral order $n \geq 0$
<code>__x</code>	The real argument $x \geq 0$

**Exceptions**

<code>std::domain_error</code>	if <code>__x &lt; 0</code> .
--------------------------------	------------------------------



**sph\_besself()**

```
float std::sph_besself (
    unsigned int __n,
    float __x) [inline]
```

Return the spherical Bessel function  $j_n(x)$  of nonnegative order `n` and `float` argument  $x \geq 0$ .

See also

`sph_bessel` for more details.

**sph\_bessell()**

```
long double std::sph_bessell (
    unsigned int __n,
    long double __x) [inline]
```

Return the spherical Bessel function  $j_n(x)$  of nonnegative order `n` and `long double` argument  $x \geq 0$ .

See also

`sph_bessel` for more details.

**sph\_legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __theta) [inline]
```

Return the spherical Legendre function of nonnegative integral degree `l` and order `m` and real angle  $\theta$  in radians. The spherical Legendre function is defined by

$$Y_l^m(\theta, \phi) = (-1)^m \left[ \frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!} \right] P_l^m(\cos \theta) \exp^{im\phi}$$

**Template Parameters**

<code>_Tp</code>	The floating-point type of the angle <code>__theta</code> .
------------------	---

**Parameters**

<code>__l</code>	The order <code>__l</code> $\geq 0$
<code>__m</code>	The degree <code>__m</code> $\geq 0$ and <code>__m</code> $\leq$ <code>__l</code>
<code>__theta</code>	The radian polar angle argument

**sph\_legendref()**

```
float std::sph_legendref (
    unsigned int __l,
    unsigned int __m,
    float __theta) [inline]
```

Return the spherical Legendre function of nonnegative integral degree `l` and order `m` and `float` angle  $\theta$  in radians.

See also

`sph_legendre` for details.

### `sph_legendre()`

```
long double std::sph_legendre (
    unsigned int __l,
    unsigned int __m,
    long double __theta) [inline]
```

Return the spherical Legendre function of nonnegative integral degree  $l$  and order  $m$  and long double angle  $\theta$  in radians.

See also

`sph_legendre` for details.

### `sph_neumann()`

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_neumann (
    unsigned int __n,
    _Tp __x) [inline]
```

Return the spherical Neumann function of integral order  $n \geq 0$  and real argument  $x \geq 0$ . The spherical Neumann function is defined by

$$n_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} N_{n+1/2}(x)$$

#### Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

#### Parameters

<code>__n</code>	The integral order $n \geq 0$
<code>__x</code>	The real argument <code>__x</code> $\geq 0$

#### Exceptions

<code>std::domain_error</code>	if <code>__x &lt; 0</code> .
--------------------------------	------------------------------

### `sph_neumannf()`

```
float std::sph_neumannf (
    unsigned int __n,
    float __x) [inline]
```

Return the spherical Neumann function of integral order  $n \geq 0$  and float argument  $x \geq 0$ .

See also

`sph_neumann` for details.

**sph\_neumannl()**

```
long double std::sph_neumannl (
    unsigned int __n,
    long double __x) [inline]
```

Return the spherical Neumann function of integral order  $n \geq 0$  and long double  $x \geq 0$ .

See also

sph\_neumann for details.

**3.11.6 Numeric Arrays**

Collaboration diagram for Numeric Arrays:

**Classes**

- class [std::gslice](#)
- class [std::gslice\\_array<\\_Tp>](#)
- class [std::indirect\\_array<\\_Tp>](#)
- class [std::mask\\_array<\\_Tp>](#)
- class [std::slice](#)
- class [std::slice\\_array<\\_Tp>](#)
- class [std::valarray<\\_Tp>](#)

**Functions**

- [std::gslice::gslice](#) ()
- [std::gslice::gslice](#) (const [gslice](#) &)
- [std::gslice::gslice](#) (size\_t \_\_o, const [valarray](#)< size\_t > &\_\_l, const [valarray](#)< size\_t > &\_\_s)
- [std::gslice\\_array<\\_Tp>::gslice\\_array](#) (const [gslice\\_array](#) &)
- [std::indirect\\_array<\\_Tp>::indirect\\_array](#) (const [indirect\\_array](#) &)
- [std::mask\\_array<\\_Tp>::mask\\_array](#) (const [mask\\_array](#) &)
- [std::slice::slice](#) ()
- [std::slice::slice](#) (size\_t \_\_o, size\_t \_\_d, size\_t \_\_s)
- [std::slice\\_array<\\_Tp>::slice\\_array](#) (const [slice\\_array](#) &)
- [std::valarray<\\_Tp>::valarray](#) () noexcept
- template<class \_Dom>  
[std::valarray<\\_Tp>::valarray](#) (const \_Expr< \_Dom, \_Tp > &\_\_e)
- [std::valarray<\\_Tp>::valarray](#) (const \_Tp &, size\_t)
- template<typename \_Tp>  
[std::valarray<\\_Tp>::valarray](#) (const \_Tp \*\_\_restrict \_\_p, size\_t \_\_n)
- [std::valarray<\\_Tp>::valarray](#) (const [gslice\\_array](#)< \_Tp > &)
- [std::valarray<\\_Tp>::valarray](#) (const [indirect\\_array](#)< \_Tp > &)

- `std::valarray<_Tp>::valarray (const mask_array<_Tp> &)`
- `std::valarray<_Tp>::valarray (const slice_array<_Tp> &)`
- `std::valarray<_Tp>::valarray (const valarray &)`
- `std::valarray<_Tp>::valarray (initializer_list<_Tp>)`
- `std::valarray<_Tp>::valarray (size_t)`
- `std::valarray<_Tp>::valarray (valarray &&) noexcept`
- `std::gslice::~gslice ()`
- `_Expr<_ValFunClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::apply (_Tp __func(_Tp)) const`
- `_Expr<_RefFunClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::apply (_Tp __func(const _Tp &)) const`
- `template<class _Tp>`  
`const _Tp * std::begin (const valarray<_Tp> &__va) noexcept`
- `template<class _Tp>`  
`_Tp * std::begin (valarray<_Tp> &__va) noexcept`
- `valarray<_Tp> std::valarray<_Tp>::cshift (int __n) const`
- `template<class _Tp>`  
`const _Tp * std::end (const valarray<_Tp> &__va) noexcept`
- `template<class _Tp>`  
`_Tp * std::end (valarray<_Tp> &__va) noexcept`
- `_Tp std::valarray<_Tp>::max () const`
- `_Tp std::valarray<_Tp>::min () const`
- `gslice & std::gslice::operator= (const gslice &)`
- `template<class _Dom>`  
`void std::gslice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `void std::gslice_array<_Tp>::operator= (const _Tp &) const`
- `gslice_array & std::gslice_array<_Tp>::operator= (const gslice_array &)`
- `void std::gslice_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void std::indirect_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `void std::indirect_array<_Tp>::operator= (const _Tp &) const`
- `indirect_array & std::indirect_array<_Tp>::operator= (const indirect_array &)`
- `void std::indirect_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `template<class _Ex>`  
`void std::mask_array<_Tp>::operator= (const _Expr<_Ex, _Tp> &__e) const`
- `void std::mask_array<_Tp>::operator= (const _Tp &) const`
- `mask_array & std::mask_array<_Tp>::operator= (const mask_array &)`
- `void std::mask_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void std::slice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator= (const _Tp &) const`
- `slice_array & std::slice_array<_Tp>::operator= (const slice_array &)`
- `void std::slice_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `template<class _Dom>`  
`valarray<_Tp> & std::valarray<_Tp>::operator= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const _Tp &__t)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const gslice_array<_Tp> &__ga)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const indirect_array<_Tp> &__ia)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const mask_array<_Tp> &__ma)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const slice_array<_Tp> &__sa)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const valarray<_Tp> &__v)`
- `valarray & std::valarray<_Tp>::operator= (initializer_list<_Tp> &__l)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (valarray<_Tp> &&__v) noexcept`

- `gslice_array<_Tp> std::valarray<_Tp>::operator[]` (const `gslice` &\_\_s)
- `_Expr<_GClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::operator[]` (const `gslice` &\_\_s) const
- `mask_array<_Tp> std::valarray<_Tp>::operator[]` (const `valarray`< bool > &\_\_m)
- `valarray<_Tp> std::valarray<_Tp>::operator[]` (const `valarray`< bool > &\_\_m) const
- `indirect_array<_Tp> std::valarray<_Tp>::operator[]` (const `valarray`< size\_t > &\_\_i)
- `_Expr<_IClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::operator[]` (const `valarray`< size\_t > &\_\_i) const
- `_Tp & std::valarray<_Tp>::operator[]` (size\_t \_\_i) noexcept
- `const _Tp & std::valarray<_Tp>::operator[]` (size\_t) const noexcept
- `slice_array<_Tp> std::valarray<_Tp>::operator[]` (slice \_\_s)
- `_Expr<_SClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::operator[]` (slice \_\_s) const
- `void std::valarray<_Tp>::resize` (size\_t \_\_size, \_Tp \_\_c=\_Tp())
- `valarray<_Tp> std::valarray<_Tp>::shift` (int \_\_n) const
- `valarray< size_t > std::gslice::size` () const
- `size_t std::slice::size` () const
- `size_t std::valarray<_Tp>::size` () const
- `size_t std::gslice::start` () const
- `size_t std::slice::start` () const
- `valarray< size_t > std::gslice::stride` () const
- `size_t std::slice::stride` () const
- `_Tp std::valarray<_Tp>::sum` () const
- `void std::valarray<_Tp>::swap` (valarray<\_Tp> &\_\_v) noexcept
- `template<typename _Tp, size_t _Nm>`  
`std::valarray` (const \_Tp(&)[\_Nm], size\_t) -> valarray<\_Tp>

### 3.11.6.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

### 3.11.6.2 Function Documentation

#### `gslice()` [1/3]

```
std::gslice::gslice () [inline]
```

Construct an empty slice.

References `gslice()`.

Referenced by `gslice()`, `gslice()`, `gslice()`, and `operator=()`.

#### `gslice()` [2/3]

```
std::gslice::gslice (
    const gslice & __g) [inline]
```

Copy constructor.

References `gslice()`.

#### `gslice()` [3/3]

```
std::gslice::gslice (
    size_t __o,
    const valarray< size_t > & __l,
    const valarray< size_t > & __s) [inline]
```

Construct a slice.

Constructs a slice with as many dimensions as the length of the `l` and `s` arrays.

## Parameters

$\leftrightarrow$ <code>_o</code>	Offset in array of first element.
$\leftrightarrow$ <code>_l</code>	Array of dimension lengths.
$\leftrightarrow$ <code>_s</code>	Array of dimension strides between array elements.

References [gslice\(\)](#).

**gslice\_array()**

```
template<typename _Tp>
std::gslice_array< _Tp >::gslice_array (
    const gslice_array< _Tp > & __a) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

References [gslice\\_array\(\)](#).

Referenced by [gslice\\_array\(\)](#), [operator=\(\)](#), and [operator>>=\(\)](#).

**indirect\_array()**

```
template<typename _Tp>
std::indirect_array< _Tp >::indirect_array (
    const indirect_array< _Tp > & __a) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

References [indirect\\_array\(\)](#).

Referenced by [indirect\\_array\(\)](#), [operator=\(\)](#), and [operator>>=\(\)](#).

**mask\_array()**

```
template<typename _Tp>
std::mask_array< _Tp >::mask_array (
    const mask_array< _Tp > & __a) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

References [mask\\_array\(\)](#).

Referenced by [mask\\_array\(\)](#), [operator=\(\)](#), and [operator>>=\(\)](#).

**slice()** [1/2]

```
std::slice::slice () [inline]
```

Construct an empty slice.

Referenced by [operator==](#).

**slice()** [2/2]

```
std::slice::slice (
    size_t __o,
    size_t __d,
    size_t __s) [inline]
```

Construct a slice.

**Parameters**

<a href="#"><code>_↔ _o</code></a>	Offset in array of first element.
<a href="#"><code>_↔ _d</code></a>	Number of elements in slice.
<a href="#"><code>_↔ _s</code></a>	Stride between array elements.

**slice\_array()**

```
template<typename _Tp>
std::slice_array< _Tp >::slice_array (
    const slice_array< _Tp > & __a) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

References [slice\\_array\(\)](#).

Referenced by [slice\\_array\(\)](#), [operator=\(\)](#), and [operator>>=\(\)](#).

**valarray()** [1/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray () [inline], [noexcept]
```

Construct an empty array.

**valarray()** [2/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const _Tp & __t,
    size_t __n) [inline]
```

Construct an array with *n* elements initialized to *t*.

**valarray()** [3/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const gslice_array< _Tp > & __ga) [inline]
```

Construct an array with the same size and values in *ga*.

**valarray()** [4/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const indirect_array< _Tp > & __ia) [inline]
```

Construct an array with the same size and values in *ia*.

**valarray()** [5/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const mask_array< _Tp > & __ma) [inline]
```

Construct an array with the same size and values in *ma*.

**valarray()** [6/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const slice_array< _Tp > & __sa) [inline]
```

Construct an array with the same size and values in *sa*.

**valarray()** [7/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const valarray< _Tp > & __v) [inline]
```

Copy constructor.

**valarray()** [8/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    initializer_list< _Tp > __l) [inline]
```

Construct an array with an *initializer\_list* of values.

**valarray()** [9/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    size_t __n) [inline], [explicit]
```

Construct an array with *n* elements.

**valarray()** [10/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    valarray< _Tp > && __v) [inline], [noexcept]
```

Move constructor.

**~gslice()**

```
std::gslice::~gslice () [inline]
```

Destructor.

**apply()** [1/2]

```
template<class _Tp>
_Expr< _ValFuncClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
    _Tp __func_Tp) const [inline]
```

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying *\_\_func* to the corresponding element of this array. The new array has the same size as this one.

**Parameters**

<i>__func</i>	Function of <i>Tp</i> returning <i>Tp</i> to apply.
---------------	---



**Returns**

New valarray with transformed elements.

**apply()** [2/2]

```
template<class _Tp>
_Expr< _RefFuncClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
    _Tp __funcconst _Tp &) const [inline]
```

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying \_\_func to the corresponding element of this array. The new array has the same size as this one.

**Parameters**

<code>__func</code>	Function of const Tp& returning Tp to apply.
---------------------	--

**Returns**

New valarray with transformed elements.

**begin()** [1/2]

```
template<class _Tp>
const _Tp * std::begin (
    const valarray< _Tp > & __va) [inline], [noexcept]
```

Return an iterator pointing to the first element of the const valarray.

**Parameters**

<code>__va</code>	valarray.
-------------------	-----------

**begin()** [2/2]

```
template<class _Tp>
_Tp * std::begin (
    valarray< _Tp > & __va) [inline], [noexcept]
```

Return an iterator pointing to the first element of the valarray.

**Parameters**

<code>__va</code>	valarray.
-------------------	-----------

Referenced by [cbegin\(\)](#), [std::vector< \\_Tp, \\_Alloc >::insert\(\)](#), [std::list< \\_Tp, \\_Alloc >::merge\(\)](#), [std::list< \\_Tp, \\_Alloc >::merge\(\)](#), [std::list< \\_Tp, \\_Alloc >::remove\(\)](#), [std::list< \\_Tp, \\_Alloc >::remove\\_if\(\)](#), [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >>::BigBlock](#), [std::list< \\_Tp, \\_Alloc >::sort\(\)](#), [std::list< \\_Tp, \\_Alloc >::sort\(\)](#), [std::list< \\_Tp, \\_Alloc >::unique\(\)](#), and [std::list< \\_Tp, \\_Alloc >::unique\(\)](#).

**cshift()**

```
template<class _Tp>
valarray< _Tp > std::valarray< _Tp >::cshift (
    int __n) const [inline]
```

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index  $i$ , the new position is  $(i - n) \% \text{size}()$ . The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

#### Parameters

<code>_↔</code>	Number of element positions to rotate.
<code>_n</code>	

#### Returns

New valarray with elements in shifted positions.

#### **end()** [1/2]

```
template<class _Tp>
const _Tp * std::end (
    const valarray< _Tp > & __va) [inline], [noexcept]
```

Return an iterator pointing to one past the last element of the const valarray.

#### Parameters

<code>__va</code>	valarray.
-------------------	-----------

#### **end()** [2/2]

```
template<class _Tp>
_Tp * std::end (
    valarray< _Tp > & __va) [inline], [noexcept]
```

Return an iterator pointing to one past the last element of the valarray.

#### Parameters

<code>__va</code>	valarray.
-------------------	-----------

Referenced by `cend()`, `std::vector< _Tp, _Alloc >::insert()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::merge()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::list< _Tp, _Alloc >::remove()`, `std::list< _Tp, _Alloc >::remove_if()`, `std::vector< _Tp, polymorphic_allocator< _Tp > >::reserve()`, `std::list< _Tp, _Alloc >::resize()`, `std::list< _Tp, _Alloc >::resize()`, `std::list< _Tp, _Alloc >::sort()`, `std::list< _Tp, _Alloc >::sort()`, `std::list< _Tp, _Alloc >::unique()`, and `std::list< _Tp, _Alloc >::unique()`.

#### **max()**

```
template<typename _Tp>
_Tp std::valarray< _Tp >::max () const [inline]
```

Return the maximum element using `operator<()`.

#### **min()**

```
template<typename _Tp>
_Tp std::valarray< _Tp >::min () const [inline]
```

Return the minimum element using `operator<()`.

**operator=()** [1/20]

```
gslice & std::gslice::operator= (
    const gslice & __g) [inline]
```

Assignment operator.

References [gslice\(\)](#).

**operator=()** [2/20]

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator= (
    const _Tp & __t) const [inline]
```

Assign all slice elements to *t*.

**operator=()** [3/20]

```
template<typename _Tp>
gslice_array< _Tp > & std::gslice_array< _Tp >::operator= (
    const gslice_array< _Tp > & __a) [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.

References [gslice\\_array\(\)](#).

**operator=()** [4/20]

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator= (
    const valarray< _Tp > & __v) const [inline]
```

Assign slice elements to corresponding elements of *v*.

References [std::valarray< \\_Tp >::size\(\)](#).

**operator=()** [5/20]

```
template<typename _Tp>
void std::indirect_array< _Tp >::operator= (
    const _Tp & __t) const [inline]
```

Assign all slice elements to *t*.

**operator=()** [6/20]

```
template<typename _Tp>
indirect_array< _Tp > & std::indirect_array< _Tp >::operator= (
    const indirect_array< _Tp > & __a) [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.

References [indirect\\_array\(\)](#).

**operator=()** [7/20]

```
template<typename _Tp>
void std::indirect_array< _Tp >::operator= (
    const valarray< _Tp > & __v) const [inline]
```

Assign slice elements to corresponding elements of *v*.

**operator=()** [8/20]

```
template<typename _Tp>
void std::mask_array< _Tp >::operator= (
    const _Tp & __t) const [inline]
```

Assign all slice elements to *t*.

**operator=()** [9/20]

```
template<typename _Tp>
mask_array< _Tp > & std::mask_array< _Tp >::operator= (
    const mask_array< _Tp > & __a) [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.  
References [mask\\_array\(\)](#).

**operator=()** [10/20]

```
template<typename _Tp>
void std::slice_array< _Tp >::operator= (
    const _Tp & __t) const [inline]
```

Assign all slice elements to *t*.

**operator=()** [11/20]

```
template<typename _Tp>
slice_array< _Tp > & std::slice_array< _Tp >::operator= (
    const slice_array< _Tp > & __a) [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.  
References [slice\\_array\(\)](#).

**operator=()** [12/20]

```
template<typename _Tp>
void std::slice_array< _Tp >::operator= (
    const valarray< _Tp > & __v) const [inline]
```

Assign slice elements to corresponding elements of *v*.

**operator=()** [13/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const _Tp & __t) [inline]
```

Assign elements to a value.  
Assign all elements of array to *t*.

**Parameters**

↵	Value for elements.
↵	
↵	
↵	
<i>t</i>	

**operator=()** [14/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const gslice_array< _Tp > & __ga) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

**Parameters**

<code>__ga</code>	Array slice to get values from.
-------------------	---------------------------------

**operator=()** [15/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const indirect_array< _Tp > & __ia) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

**Parameters**

<code>__↵ _ia</code>	Array slice to get values from.
--------------------------	---------------------------------

**operator=()** [16/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const mask_array< _Tp > & __ma) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

**Parameters**

<code>__ma</code>	Array slice to get values from.
-------------------	---------------------------------

**operator=()** [17/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const slice_array< _Tp > & __sa) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

**Parameters**

<code>__sa</code>	Array slice to get values from.
-------------------	---------------------------------

**operator=()** [18/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const valarray< _Tp > & __v) [inline]
```

Assign elements to an array.

Assign elements of array to values in *v*.

**Parameters**

<code>__v</code>	Valarray to get values from.
------------------	------------------------------

**operator=()** [19/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    initializer_list< _Tp > __l) [inline]
```

Assign elements to an initializer\_list.

Assign elements of array to values in `__l`. Results are undefined if `__l` does not have the same size as this array.

**Parameters**

<code>__l</code>	initializer_list to get values from.
------------------	--------------------------------------

**operator=()** [20/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    valarray< _Tp > && __v) [inline], [noexcept]
```

Move assign elements to an array.

Move assign elements of array to values in *v*.

**Parameters**

<code>__v</code>	Valarray to get values from.
------------------	------------------------------

**operator[]()** [1/9]

```
template<typename _Tp>
gslice_array< _Tp > std::valarray< _Tp >::operator[] (
    const gslice & __s) [inline]
```

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the gslice argument. The new valarray has the same size as the input gslice.

**See also**

gslice.

**Parameters**

$\_ \leftrightarrow$	The source gslice.
$\_s$	

**Returns**

New valarray containing elements in  $\_s$ .

**operator[]() [2/9]**

```
template<typename _Tp>
_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
    const gslice & __s) const [inline]
```

Return an array subset.

Returns a slice\_array referencing the elements of the array indicated by the slice argument.

**See also**

gslice.

**Parameters**

$\_ \leftrightarrow$	The source slice.
$\_s$	

**Returns**

Slice\_array referencing elements indicated by  $\_s$ .

**operator[]() [3/9]**

```
template<typename _Tp>
mask_array< _Tp > std::valarray< _Tp >::operator[] (
    const valarray< bool > & __m) [inline]
```

Return a reference to an array subset.

Returns a new mask\_array referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

**Parameters**

$\_ \leftrightarrow$	The valarray bitmask.
$\_m$	

**Returns**

New valarray containing elements indicated by  $\_m$ .

**operator[]()** [4/9]

```
template<typename _Tp>
valarray< _Tp > std::valarray< _Tp >::operator[] (
    const valarray< bool > & __m) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

**Parameters**

$\leftarrow$	The valarray bitmask.
$\_m$	

**Returns**

New valarray containing elements indicated by  $\_m$ .

**operator[]()** [5/9]

```
template<typename _Tp>
indirect_array< _Tp > std::valarray< _Tp >::operator[] (
    const valarray< size_t > & __i) [inline]
```

Return a reference to an array subset.

Returns an indirect\_array referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned indirect\_array refers to these elements.

**Parameters**

$\leftarrow$	The valarray element index list.
$\_ \leftarrow$	
$\leftarrow$	
$\_ \leftarrow$	
$i$	

**Returns**

Indirect\_array referencing elements in  $\_i$ .

**operator[]()** [6/9]

```
template<typename _Tp>
_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
    const valarray< size_t > & __i) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.



**Parameters**

	The valarray element index list.
	
	
	
	

**Returns**

New valarray containing elements in `__s`.

**operator[]() [7/9]**

```
template<typename _Tp>
_Tp & std::valarray< _Tp >::operator[] (
    size_t __i) [inline], [noexcept]
```

Return a reference to the *i*'th array element.

**Parameters**

	Index of element to return.
	
	
	
	

**Returns**

Reference to the *i*'th element.

**operator[]() [8/9]**

```
template<typename _Tp>
slice_array< _Tp > std::valarray< _Tp >::operator[] (
    slice __s) [inline]
```

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

**See also**

`slice`.

**Parameters**

	The source slice.
	

**Returns**

New valarray containing elements in `__s`.

**operator[]()** [9/9]

```
template<typename _Tp>
_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
    slice __s) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

**See also**

slice.

**Parameters**

<code>__s</code>	The source slice.
------------------	-------------------

**Returns**

New valarray containing elements in `__s`.

**resize()**

```
template<class _Tp>
void std::valarray< _Tp >::resize (
    size_t __size,
    _Tp __c = _Tp()) [inline]
```

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

**Parameters**

<code>__size</code>	New array size.
<code>__c</code>	New value for all elements.

**shift()**

```
template<class _Tp>
valarray< _Tp > std::valarray< _Tp >::shift (
    int __n) const [inline]
```

Return a shifted array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is *i* - *n*. The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements [0, *n*). Negative arguments discard elements from the top of the array.

**Parameters**

<code>__n</code>	Number of element positions to shift.
------------------	---------------------------------------

**Returns**

New valarray with elements in shifted positions.

**size()** [1/3]

```
valarray< size_t > std::gslice::size () const [inline]
```

Return array of sizes of slice dimensions.

**size()** [2/3]

```
size_t std::slice::size () const [inline]
```

Return size of slice.

**size()** [3/3]

```
template<class _Tp>
```

```
size_t std::valarray< _Tp >::size () const [inline]
```

Return the number of elements in array.

Referenced by [std::gslice\\_array< \\_Tp >::operator=\(\)](#).

**start()** [1/2]

```
size_t std::gslice::start () const [inline]
```

Return array offset of first slice element.

**start()** [2/2]

```
size_t std::slice::start () const [inline]
```

Return array offset of first slice element.

**stride()** [1/2]

```
valarray< size_t > std::gslice::stride () const [inline]
```

Return array of array strides for each dimension.

**stride()** [2/2]

```
size_t std::slice::stride () const [inline]
```

Return array stride of slice.

**sum()**

```
template<class _Tp>
```

```
_Tp std::valarray< _Tp >::sum () const [inline]
```

Return the sum of all elements in the array.

Accumulates the sum of all elements into a `Tp` using `+=`. The order of adding the elements is unspecified.

**swap()**

```
template<class _Tp>
```

```
void std::valarray< _Tp >::swap (  
    valarray< _Tp > & __v) [inline], [noexcept]
```

Swap.

### 3.11.7 Random Number Generation

Collaboration diagram for Random Number Generation:



#### Topics

- [Random Number Distributions](#)
- [Random Number Generators](#)
- [Random Number Utilities](#)

#### Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>  
_RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`

#### 3.11.7.1 Detailed Description

A facility for generating random numbers on selected distributions.

#### 3.11.7.2 Function Documentation

##### **generate\_canonical()**

```
template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
_RealType std::generate_canonical (
    _UniformRandomNumberGenerator & __g)
```

A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

References [std::\\_\\_numeric\\_limits\\_base::digits](#), [std::numeric\\_limits< \\_Tp >::epsilon\(\)](#), [log\(\)](#), [max\(\)](#), and [min\(\)](#).

### 3.11.7.3 Random Number Distributions

Collaboration diagram for Random Number Distributions:



#### Topics

- [Bernoulli Distributions](#)
- [Normal Distributions](#)
- [Poisson Distributions](#)
- [Sampling Distributions](#)
- [Uniform Distributions](#)

#### 3.11.7.3.1 Detailed Description

#### 3.11.7.3.2 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



#### Classes

- class `std::bernoulli_distribution`
- class `std::binomial_distribution<_IntType>`
- class `std::geometric_distribution<_IntType>`
- class `std::negative_binomial_distribution<_IntType>`

#### Functions

- template<typename `_CharT`, typename `_Traits`>  
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> & __os, const std::bernoulli_distribution & __x)`

- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::geometric_distribution< _IntType > &__x)`

## Detailed Description

### Function Documentation

#### `operator<<()` [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::bernoulli_distribution & __x)
```

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>bernoulli_distribution</code> random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### `operator<<()` [2/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::geometric_distribution< _IntType > & __x)
```

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>geometric_distribution</code> random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

References `std::numeric_limits< _Tp >::epsilon()`, `log()`, and `std::numeric_limits< _Tp >::max()`.

**operator>>()** [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::bernoulli_distribution & __x) [inline]
```

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>bernoulli_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

References [std::bernoulli\\_distribution::param\(\)](#).

**operator>>()** [2/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::geometric_distribution< _IntType > & __x)
Extracts a geometric_distribution random number distribution __x from the input stream __is.
```

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>geometric_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

References [std::ios\\_base::flags\(\)](#).

**3.11.7.3.3 Normal Distributions**

Collaboration diagram for Normal Distributions:



## Classes

- class [std::cauchy\\_distribution< \\_RealType >](#)
- class [std::chi\\_squared\\_distribution< \\_RealType >](#)
- class [std::fisher\\_f\\_distribution< \\_RealType >](#)
- class [std::lognormal\\_distribution< \\_RealType >](#)
- class [std::normal\\_distribution< \\_RealType >](#)
- class [std::student\\_t\\_distribution< \\_RealType >](#)



## Functions

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::cauchy_distribution< _RealType > &__x)`

## Detailed Description

### Function Documentation

#### `operator<<()`

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::cauchy_distribution< _RealType > & __x)
```

Inserts a `cauchy_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>cauchy_distribution</code> random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

References [tan\(\)](#).

#### `operator>>()`

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::cauchy_distribution< _RealType > & __x)
```

Extracts a `cauchy_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>cauchy_distribution</code> random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::cauchy\\_distribution< \\_RealType >::param\(\)](#).

### 3.11.7.3.4 Poisson Distributions

Collaboration diagram for Poisson Distributions:



#### Classes

- class `std::exponential_distribution< _RealType >`
- class `std::extreme_value_distribution< _RealType >`
- class `std::gamma_distribution< _RealType >`
- class `std::poisson_distribution< _IntType >`
- class `std::weibull_distribution< _RealType >`

#### Functions

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)`

#### Detailed Description

#### Function Documentation

##### `operator<<()` [1/3]

```

template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::exponential_distribution< _RealType > & __x)

```

Inserts a `exponential_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>exponential_distribution</code> random number distribution.

## Returns

The output stream with the state of `__x` inserted or in an error state.

References [log\(\)](#).

**operator<<()** [2/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::extreme_value_distribution< _RealType > & __x)
```

Inserts a `extreme_value_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>extreme_value_distribution</code> random number distribution.

## Returns

The output stream with the state of `__x` inserted or in an error state.

References [log\(\)](#).

**operator<<()** [3/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::weibull_distribution< _RealType > & __x)
```

Inserts a `weibull_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>weibull_distribution</code> random number distribution.

## Returns

The output stream with the state of `__x` inserted or in an error state.

References [log\(\)](#), and [pow\(\)](#).

**operator>>()** [1/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::exponential_distribution< _RealType > & __x)
```

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

$\_is$	An input stream.
$\_x$	A exponential_distribution random number generator engine.

## Returns

The input stream with  $\_x$  extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::exponential\\_distribution<\\_RealType>::param\(\)](#).

**operator>>()** [2/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::extreme_value_distribution< _RealType > & __x)
```

Extracts a extreme\_value\_distribution random number distribution  $\_x$  from the input stream  $\_is$ .

## Parameters

$\_is$	An input stream.
$\_x$	A extreme_value_distribution random number generator engine.

## Returns

The input stream with  $\_x$  extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::extreme\\_value\\_distribution<\\_RealType>::param\(\)](#).

**operator>>()** [3/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::weibull_distribution< _RealType > & __x)
```

Extracts a weibull\_distribution random number distribution  $\_x$  from the input stream  $\_is$ .

## Parameters

$\_is$	An input stream.
$\_x$	A weibull_distribution random number generator engine.

## Returns

The input stream with  $\_x$  extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::weibull\\_distribution<\\_RealType>::param\(\)](#).

### 3.11.7.3.5 Sampling Distributions

Collaboration diagram for Sampling Distributions:



#### Classes

- class [std::discrete\\_distribution<\\_IntType>](#)
- class [std::piecewise\\_constant\\_distribution<\\_RealType>](#)
- class [std::piecewise\\_linear\\_distribution<\\_RealType>](#)

#### Detailed Description

### 3.11.7.3.6 Uniform Distributions

Collaboration diagram for Uniform Distributions:



#### Classes

- class [std::uniform\\_int\\_distribution<\\_IntType>](#)
- class [std::uniform\\_real\\_distribution<\\_RealType>](#)

#### Functions

- [template<typename \\_IntType, typename \\_CharT, typename \\_Traits>](#)  
[std::basic\\_ostream<\\_CharT, \\_Traits>](#) & [std::operator<<](#) ([std::basic\\_ostream<\\_CharT, \\_Traits>](#) &, const [std::uniform\\_int\\_distribution<\\_IntType>](#) &)
- [template<typename \\_RealType, typename \\_CharT, typename \\_Traits>](#)  
[std::basic\\_ostream<\\_CharT, \\_Traits>](#) & [std::operator<<](#) ([std::basic\\_ostream<\\_CharT, \\_Traits>](#) &, const [std::uniform\\_real\\_distribution<\\_RealType>](#) &)
- [template<typename \\_IntType, typename \\_CharT, typename \\_Traits>](#)  
[std::basic\\_istream<\\_CharT, \\_Traits>](#) & [std::operator>>](#) ([std::basic\\_istream<\\_CharT, \\_Traits>](#) &, [std::uniform\\_int\\_distribution<\\_IntType>](#) &)
- [template<typename \\_RealType, typename \\_CharT, typename \\_Traits>](#)  
[std::basic\\_istream<\\_CharT, \\_Traits>](#) & [std::operator>>](#) ([std::basic\\_istream<\\_CharT, \\_Traits>](#) &, [std::uniform\\_real\\_distribution<\\_RealType>](#) &)

## Detailed Description

### Function Documentation

#### **operator<<()** [1/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::uniform_int_distribution< _IntType > & __x)
```

Inserts a `uniform_int_distribution` random number distribution `__x` into the output stream `os`.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>uniform_int_distribution</code> random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

References `std::ios_base::flags()`.

#### **operator<<()** [2/2]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::uniform_real_distribution< _RealType > & __x)
```

Inserts a `uniform_real_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>uniform_real_distribution</code> random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### **operator>>()** [1/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::uniform_int_distribution< _IntType > & __x)
```

Extracts a `uniform_int_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>uniform_int_distribution</code> random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::uniform\\_int\\_distribution<\\_IntType>::param\(\)](#).

**operator>>()** [2/2]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & std::operator>> (
    std::basic_istream<_CharT, _Traits> & __is,
    std::uniform_real_distribution<_RealType> & __x)

```

Extracts a `uniform_real_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A <code>uniform_real_distribution</code> random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::uniform\\_real\\_distribution<\\_RealType>::param\(\)](#).

**3.11.7.4 Random Number Generators**

Collaboration diagram for Random Number Generators:

**Classes**

- class [std::discard\\_block\\_engine<\\_RandomNumberEngine, \\_\\_p, \\_\\_r>](#)
- class [std::independent\\_bits\\_engine<\\_RandomNumberEngine, \\_\\_w, \\_UIntType>](#)
- class [std::linear\\_congruential\\_engine<\\_UIntType, \\_\\_a, \\_\\_c, \\_\\_m>](#)
- class [std::mersenne\\_twister\\_engine<\\_UIntType, \\_\\_w, \\_\\_n, \\_\\_m, \\_\\_r, \\_\\_a, \\_\\_u, \\_\\_d, \\_\\_s, \\_\\_b, \\_\\_t, \\_\\_c, \\_\\_l, \\_\\_f>](#)
- class [std::random\\_device](#)
- class [std::shuffle\\_order\\_engine<\\_RandomNumberEngine, \\_\\_k>](#)
- class [std::subtract\\_with\\_carry\\_engine<\\_UIntType, \\_\\_w, \\_\\_s, \\_\\_r>](#)

**Typedefs**

- typedef [minstd\\_rand0](#) [std::default\\_random\\_engine](#)
- typedef [shuffle\\_order\\_engine<minstd\\_rand0, 256>](#) [std::knuth\\_b](#)
- typedef [linear\\_congruential\\_engine<uint\\_fast32\\_t, 48271UL, 0UL, 2147483647UL>](#) [std::minstd\\_rand](#)

- typedef `linear_congruential_engine`< uint\_fast32\_t, 16807UL, 0UL, 2147483647UL > `std::minstd_rand0`
- typedef `mersenne_twister_engine`< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > `std::mt19937`
- typedef `mersenne_twister_engine`< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL > `std::mt19937_64`
- typedef `discard_block_engine`< `ranlux24_base`, 223, 23 > `std::ranlux24`
- typedef `subtract_with_carry_engine`< uint\_fast32\_t, 24, 10, 24 > `std::ranlux24_base`
- typedef `discard_block_engine`< `ranlux48_base`, 389, 11 > `std::ranlux48`
- typedef `subtract_with_carry_engine`< uint\_fast64\_t, 48, 5, 12 > `std::ranlux48_base`

## Functions

- template<typename \_RandomNumberEngine, size\_t \_\_w, typename \_UIntType, typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &\_\_os, const  
`std::independent_bits_engine`< \_RandomNumberEngine, \_\_w, \_UIntType > &\_\_x)

### 3.11.7.4.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an operator() that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

**Table 291 Random Number Generator Requirements**

To be documented.
-------------------

### 3.11.7.4.2 Typedef Documentation

#### `minstd_rand`

```
typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand
```

An alternative LCR (Lehmer Generator function).

#### `minstd_rand0`

```
typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0
```

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

#### `mt19937`

```
typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937
```

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

#### `mt19937_64`

```
typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL> std::mt19937_64
```

An alternative Mersenne Twister.



### 3.11.7.4.3 Function Documentation

#### **operator<<()**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename
_traits>
```

```
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x)
```

Inserts the current state of a `independent_bits_engine` random number generator engine `__x` into the output stream `__os`.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>independent_bits_engine</code> random number generator engine.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

References [std::\\_\\_numeric\\_limits\\_base::digits](#).

### 3.11.7.5 Random Number Utilities

Collaboration diagram for Random Number Utilities:



#### Classes

- class [std::seed\\_seq](#)

#### 3.11.7.5.1 Detailed Description

### 3.11.8 TR1 Mathematical Special Functions

Collaboration diagram for TR1 Mathematical Special Functions:



## Functions

- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre` (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_laguerref` (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_laguerrel` (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tpx, typename _Tpy>`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (\_Tpx \_\_x, \_Tpy \_\_y)
- `float std::tr1::betaf` (float \_\_x, float \_\_y)
- `long double std::tr1::betal` (long double \_\_x, long double \_\_y)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1` (\_Tp \_\_k)
- `float std::tr1::comp_ellint_1f` (float \_\_k)
- `long double std::tr1::comp_ellint_1l` (long double \_\_k)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2` (\_Tp \_\_k)
- `float std::tr1::comp_ellint_2f` (float \_\_k)
- `long double std::tr1::comp_ellint_2l` (long double \_\_k)
- `template<typename _Tp, typename _Tpn>`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3` (\_Tp \_\_k, \_Tpn \_\_nu)
- `float std::tr1::comp_ellint_3f` (float \_\_k, float \_\_nu)
- `long double std::tr1::comp_ellint_3l` (long double \_\_k, long double \_\_nu)
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_bessel_if` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_bessel_il` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_bessel_jf` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_bessel_jl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_bessel_kf` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_bessel_kl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_neumannf` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_neumannl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1` (\_Tp \_\_k, \_Tpp \_\_phi)
- `float std::tr1::ellint_1f` (float \_\_k, float \_\_phi)
- `long double std::tr1::ellint_1l` (long double \_\_k, long double \_\_phi)
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2` (\_Tp \_\_k, \_Tpp \_\_phi)
- `float std::tr1::ellint_2f` (float \_\_k, float \_\_phi)
- `long double std::tr1::ellint_2l` (long double \_\_k, long double \_\_phi)
- `template<typename _Tp, typename _Tpn, typename _Tpp>`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3` (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)

- float **std::tr1::ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **std::tr1::ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::expint** (\_Tp \_\_x)
- float **std::tr1::expintf** (float \_\_x)
- long double **std::tr1::expintl** (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::hermitef** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::laguerref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::legendre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::legendref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::legendrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::riemann\_zeta** (\_Tp \_\_x)
- float **std::tr1::riemann\_zetaf** (float \_\_x)
- long double **std::tr1::riemann\_zetal** (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_bessel** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **std::tr1::sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **std::tr1::sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

### 3.11.8.1 Detailed Description

A collection of advanced mathematical special functions.

### 3.11.8.2 Function Documentation

#### assoc\_laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (
    unsigned int __n,
    unsigned int __m,
    _Tp __x) [inline]
```

5.2.1.1 Associated Laguerre polynomials.

**assoc\_legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __x) [inline]
```

## 5.2.1.2 Associated Legendre functions.

**beta()**

```
template<typename _Tpx, typename _Tpy>
__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (
    _Tpx __x,
    _Tpy __y) [inline]
```

## 5.2.1.3 Beta functions.

**comp\_ellint\_1()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (
    _Tp __k) [inline]
```

## 5.2.1.4 Complete elliptic integrals of the first kind.

**comp\_ellint\_2()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (
    _Tp __k) [inline]
```

## 5.2.1.5 Complete elliptic integrals of the second kind.

**comp\_ellint\_3()**

```
template<typename _Tp, typename _Tpn>
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3 (
    _Tp __k,
    _Tpn __nu) [inline]
```

## 5.2.1.6 Complete elliptic integrals of the third kind.

**cyl\_bessel\_i()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i (
    _Tpnu __nu,
    _Tp __x) [inline]
```

## 5.2.1.8 Regular modified cylindrical Bessel functions.

**cyl\_bessel\_j()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j (
    _Tpnu __nu,
    _Tp __x) [inline]
```

## 5.2.1.9 Cylindrical Bessel functions (of the first kind).

**cyl\_bessel\_k()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (
    _Tpnu __nu,
    _Tp __x) [inline]
```

5.2.1.10 Irregular modified cylindrical Bessel functions.

**cyl\_neumann()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (
    _Tpnu __nu,
    _Tp __x) [inline]
```

5.2.1.11 Cylindrical Neumann functions.

**ellint\_1()**

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (
    _Tp __k,
    _Tpp __phi) [inline]
```

5.2.1.12 Incomplete elliptic integrals of the first kind.

**ellint\_2()**

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (
    _Tp __k,
    _Tpp __phi) [inline]
```

5.2.1.13 Incomplete elliptic integrals of the second kind.

**ellint\_3()**

```
template<typename _Tp, typename _Tpn, typename _Tpp>
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (
    _Tp __k,
    _Tpn __nu,
    _Tpp __phi) [inline]
```

5.2.1.14 Incomplete elliptic integrals of the third kind.

**expint()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (
    _Tp __x) [inline]
```

5.2.1.15 Exponential integrals.

**hermite()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (
    unsigned int __n,
    _Tp __x) [inline]
```

5.2.1.16 Hermite polynomials.

**laguerre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::laguerre (
    unsigned int __n,
    _Tp __x) [inline]
```

5.2.1.18 Laguerre polynomials.

**legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::legendre (
    unsigned int __n,
    _Tp __x) [inline]
```

5.2.1.19 Legendre polynomials.

**riemann\_zeta()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::riemann_zeta (
    _Tp __x) [inline]
```

5.2.1.20 Riemann zeta function.

**sph\_bessel()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::sph_bessel (
    unsigned int __n,
    _Tp __x) [inline]
```

5.2.1.21 Spherical Bessel functions.

**sph\_legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::sph_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __theta) [inline]
```

5.2.1.22 Spherical associated Legendre functions.

**sph\_neumann()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::sph_neumann (
    unsigned int __n,
    _Tp __x) [inline]
```

5.2.1.23 Spherical Neumann functions.

### 3.12 Regular Expressions

Collaboration diagram for Regular Expressions:



#### Topics

- [Base and Implementation Classes](#)

#### Namespaces

- namespace [std::regex\\_constants](#)

#### Classes

- class [std::basic\\_regex](#)< [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >
- class [std::match\\_results](#)< [\\_Bi\\_iter](#), [\\_Alloc](#) >
- class [std::regex\\_error](#)
- class [std::regex\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >
- class [std::regex\\_token\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >
- class [std::regex\\_traits](#)< [\\_Ch\\_type](#) >
- class [std::sub\\_match](#)< [\\_Bilter](#) >

#### Typedefs

- typedef [match\\_results](#)< const char \* > **std::cmatch**
- typedef [regex\\_iterator](#)< const char \* > **std::cregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const char \* > **std::cregex\_token\_iterator**
- typedef [sub\\_match](#)< const char \* > **std::csub\_match**
- typedef [basic\\_regex](#)< char > **std::regex**
- typedef [match\\_results](#)< string::const\_iterator > **std::smatch**
- typedef [regex\\_iterator](#)< string::const\_iterator > **std::sregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< string::const\_iterator > **std::sregex\_token\_iterator**
- typedef [sub\\_match](#)< string::const\_iterator > **std::ssub\_match**
- typedef [match\\_results](#)< const wchar\_t \* > **std::wcmatch**
- typedef [regex\\_iterator](#)< const wchar\_t \* > **std::wcregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const wchar\_t \* > **std::wcregex\_token\_iterator**
- typedef [sub\\_match](#)< const wchar\_t \* > **std::wcs\_sub\_match**
- typedef [basic\\_regex](#)< wchar\_t > **std::wregex**
- typedef [match\\_results](#)< wstring::const\_iterator > **std::wsmatch**
- typedef [regex\\_iterator](#)< wstring::const\_iterator > **std::wsregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< wstring::const\_iterator > **std::wsregex\_token\_iterator**
- typedef [sub\\_match](#)< wstring::const\_iterator > **std::wssub\_match**

## Functions

- `template<typename _ForwardIterator>`  
`std::basic_regex` (`_ForwardIterator`, `_ForwardIterator`, `regex_constants::syntax_option_type`={}) -> `basic_regex< typename iterator_traits< _ForwardIterator >::value_type >`
- `template<typename _Bi_iter, typename _Alloc>`  
`bool operator==` (`const match_results< _Bi_iter, _Alloc > &__m1`, `const match_results< _Bi_iter, _Alloc > &__m2`)
- `template<typename _Ch_type, typename _Rx_traits>`  
`void swap` (`basic_regex< _Ch_type, _Rx_traits > &__lhs`, `basic_regex< _Ch_type, _Rx_traits > &__rhs`) `noexcept`
- `template<typename _Bi_iter, typename _Alloc>`  
`void swap` (`match_results< _Bi_iter, _Alloc > &__lhs`, `match_results< _Bi_iter, _Alloc > &__rhs`) `noexcept`

## Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`_Bi_iter __s`, `_Bi_iter __e`, `match_results< _Bi_iter, _Alloc > &__m`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`_Bi_iter __first`, `_Bi_iter __last`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits>`  
`bool std::regex_match` (`const _Ch_type *__s`, `match_results< const _Ch_type *, _Alloc > &__m`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __f=regex_constants::match_default`)
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s`, `match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&`, `match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &`, `const basic_regex< _Ch_type, _Rx_traits > &`, `regex_constants::match_flag_type=regex_constants::match_default`) `=delete`
- `template<typename _Ch_type, class _Rx_traits>`  
`bool std::regex_match` (`const _Ch_type *__s`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __f=regex_constants::match_default`)
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search` (`_Bi_iter __s`, `_Bi_iter __e`, `match_results< _Bi_iter, _Alloc > &__m`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search` (`_Bi_iter __first`, `_Bi_iter __last`, `const basic_regex< _Ch_type, _Rx_traits > &__re`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)
- `template<typename _Ch_type, class _Alloc, class _Rx_traits>`  
`bool std::regex_search` (`const _Ch_type *__s`, `match_results< const _Ch_type *, _Alloc > &__m`, `const basic_regex< _Ch_type, _Rx_traits > &__e`, `regex_constants::match_flag_type __f=regex_constants::match_default`)
- `template<typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search` (`const _Ch_type *__s`, `const basic_regex< _Ch_type, _Rx_traits > &__e`, `regex_constants::match_flag_type __f=regex_constants::match_default`)
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search` (`const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s`, `const basic_regex< _Ch_type, _Rx_traits > &__e`, `regex_constants::match_flag_type __flags=regex_constants::match_default`)



- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type,`  
`_Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↵`  
`flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, ↵`  
`_Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__↵`  
`s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const`  
`basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __↵`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`basic_string< _Ch_type > std::regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx↵`  
`_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↵`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type>`  
`basic_string< _Ch_type > std::regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits`  
`> &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bilter>`  
`bool operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bilter>`  
`auto operator<=> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs) noexcept(__↵`  
`detail::__is_contiguous_iter< _Bilter >::value)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch↵`  
`_alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, ↵`  
`_Alloc > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Bi_iter>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`* __rhs)`
- `template<typename _Bi_iter>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`* __rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Bi_iter>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`& __rhs)`
- `template<typename _Bi_iter>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`& __rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`

- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>`  
`basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > & __os, const`  
`sub_match< _Bi_iter > & __m)`

### 3.12.1 Detailed Description

A facility for performing regular expression pattern matching.

Since

C++11

### 3.12.2 Typedef Documentation

#### **cregex\_token\_iterator**

typedef `regex_token_iterator<const char*>` `std::cregex_token_iterator`

Token iterator for C-style NULL-terminated strings.

#### **csub\_match**

typedef `sub_match<const char*>` `std::csub_match`

Standard regex submatch over a C-style null-terminated string.

#### **regex**

typedef `basic_regex<char>` `std::regex`

Standard regular expressions.

#### **sregex\_token\_iterator**

typedef `regex_token_iterator<string::const_iterator>` `std::sregex_token_iterator`

Token iterator for standard strings.

#### **ssub\_match**

typedef `sub_match<string::const_iterator>` `std::ssub_match`

Standard regex submatch over a standard string.

#### **wcregex\_token\_iterator**

typedef `regex_token_iterator<const wchar_t*>` `std::wcregex_token_iterator`

Token iterator for C-style NULL-terminated wide strings.

#### **wcsub\_match**

typedef `sub_match<const wchar_t*>` `std::wcsub_match`

Regex submatch over a C-style null-terminated wide string.

#### **wregex**

typedef `basic_regex<wchar_t>` `std::wregex`

Standard wide-character regular expressions.

#### **wsregex\_token\_iterator**

typedef `regex_token_iterator<wstring::const_iterator>` `std::wsregex_token_iterator`

Token iterator for standard wide-character strings.

**wssub\_match**

typedef [sub\\_match](#)<wstring::const\_iterator> [std::wssub\\_match](#)  
 Regex submatch over a standard wide string.

**3.12.3 Function Documentation****operator<<()**

```
template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>
basic\_ostream< _Ch_type, _Ch_traits > & operator<< (
    basic\_ostream< _Ch_type, _Ch_traits > & __os,
    const sub\_match< _Bi_iter > & __m) [related]
```

Inserts a matched string into an output stream.

**Parameters**

<code>__os</code>	The output stream.
<code>__m</code>	A submatch string.

**Returns**

the output stream with the submatch string inserted.

**operator<=>() [1/4]**

```
template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>
auto operator<=> (
    const sub\_match< _Bi_iter > & __lhs,
    const \_\_sub\_match\_string< _Bi_iter, _Ch_traits, _Alloc > & __rhs) [related]
```

Three-way comparison of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator<=>() [2/4]**

```
template<typename _Bi_iter>
auto operator<=> (
    const sub\_match< _Bi_iter > & __lhs,
    typename iterator\_traits< _Bi_iter >::value_type const & __rhs) [related]
```

Three-way comparison of a regular expression submatch and a character.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A character.

**Returns**

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator<=>() [3/4]**

```
template<typename _Bi_iter>
auto operator<=> (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs) [related]
```

Three-way comparison of a regular expression submatch and a C string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A null-terminated string.

**Returns**

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator<=>() [4/4]**

```
template<typename _BiIter>
auto operator<=> (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs) [related]
```

Three-way comparison of two regular expression submatches.

**Parameters**

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

**Returns**

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator==( ) [1/5]**

```
template<typename _Bi_iter, typename _Alloc>
bool operator== (
    const match_results< _Bi_iter, _Alloc > & __m1,
    const match_results< _Bi_iter, _Alloc > & __m2) [related]
```

Compares two `match_results` for equality.

**Returns**

true if the two objects refer to the same match, false otherwise.

**operator==(** [2/5]

```
template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>
bool operator== (
    const sub\_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs) [related]
```

Tests the equivalence of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

**operator==(** [3/5]

```
template<typename _Bi_iter>
bool operator== (
    const sub\_match< _Bi_iter > & __lhs,
    typename iterator\_traits< _Bi_iter >::value_type const & __rhs) [related]
```

Tests the equivalence of a regular expression submatch and a character.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A character.

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

**operator==(** [4/5]

```
template<typename _Bi_iter>
bool operator== (
    const sub\_match< _Bi_iter > & __lhs,
    typename iterator\_traits< _Bi_iter >::value_type const * __rhs) [related]
```

Tests the equivalence of a regular expression submatch and a C string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A null-terminated string.

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

**operator==()** [5/5]

```
template<typename _BiIter>
bool operator== (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs) [related]
```

Tests the equivalence of two regular expression submatches.

**Parameters**

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

**regex\_match()** [1/7]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Indicates if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

**Parameters**

<code>__first</code>	Beginning of the character sequence to match.
<code>__last</code>	One-past-the-end of the character sequence to match.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

**Return values**

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

**Exceptions**

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

References [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

**regex\_match()** [2/7]

```
template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
    _Bi_iter __s,
```

```

    _Bi_iter __e,
    match_results< _Bi_iter, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]

```

Determines if there is a match between the regular expression `e` and all of the character sequence [first, last).

#### Parameters

<code>__s</code>	Start of the character sequence to match.
<code>__e</code>	One-past-the-end of the character sequence to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

#### Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

#### Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

References [std::regex\\_constants::match\\_default](#).

Referenced by [regex\\_match\(\)](#), [regex\\_match\(\)](#), [regex\\_match\(\)](#), [regex\\_match\(\)](#), and [regex\\_match\(\)](#).

#### **regex\_match()** [3/7]

```

template<typename _Ch_type, class _Rx_traits>
bool std::regex_match (
    const _Ch_type * __s,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __f = regex_constants::match_default) [inline]

```

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

#### Parameters

<code>__s</code>	The C-style null-terminated string to match.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

#### Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

**regex\_match()** [4/7]

```
template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
bool std::regex_match (
    const _Ch_type * __s,
    match_results< const _Ch_type *, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __f = regex_constants::match_default) [inline]
```

Determines if there is a match between the regular expression `e` and a C-style null-terminated string.

## Parameters

<code>__s</code>	The C-style null-terminated string to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

## Return values

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

**regex\_match()** [5/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & ,
    const basic_regex< _Ch_type, _Rx_traits > & ,
    regex_constants::match_flag_type = regex_constants::match_default) [delete]
```

Prevent unsafe attempts to get `match_results` from a temporary string.

References [std::regex\\_constants::match\\_default](#).



**regex\_match()** [6/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↵
Rx_traits>
bool std::regex_match (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↵
iterator, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Determines if there is a match between the regular expression *e* and a string.

**Parameters**

<i>__s</i>	The string to match.
<i>__m</i>	The match results.
<i>__re</i>	The regular expression.
<i>__flags</i>	Controls how the regular expression is matched.

**Return values**

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

**Exceptions**

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::begin\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

**regex\_match()** [7/7]

```
template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
    const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Indicates if there is a match between the regular expression *e* and a string.

**Parameters**

<i>__s</i>	[IN] The string to match.
<i>__re</i>	[IN] The regular expression.
<i>__flags</i>	[IN] Controls how the regular expression is matched.

**Return values**

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::begin\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

**regex\_replace()** [1/6]

```
template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
_Out_iter std::regex_replace (
    _Out_iter __out,
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex<_Ch_type, _Rx_traits > & __e,
    const _Ch_type * __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default)
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format C-string.

## Parameters

<code>__out</code>	[OUT] The output iterator.
<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

## Returns

`__out`

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::regex\\_constants::match\\_default](#).

**regex\_replace()** [2/6]

```
template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename
_St, typename _Sa>
_Out_iter std::regex_replace (
    _Out_iter __out,
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex<_Ch_type, _Rx_traits > & __e,
    const basic_string<_Ch_type, _St, _Sa > & __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format string.

## Parameters

<code>__out</code>	[OUT] The output iterator.
<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

## Returns

`__out`

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::length\(\)](#), and [std::regex\\_constants::match\\_default](#).

Referenced by [regex\\_replace\(\)](#), [regex\\_replace\(\)](#), [regex\\_replace\(\)](#), and [regex\\_replace\(\)](#).

**regex\_replace()** [3/6]

```
template<typename _Rx_traits, typename _Ch_type>
basic_string<_Ch_type> std::regex_replace (
    const _Ch_type * __s,
    const basic_regex<_Ch_type, _Rx_traits> & __e,
    const _Ch_type * __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format C-string.

## Parameters

<code>__s</code>	[IN] The C-string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

## Returns

The string after replacing.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [back\\_inserter\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_replace\(\)](#).

**regex\_replace()** [4/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
basic_string< _Ch_type > std::regex_replace (
    const _Ch_type * __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    const basic_string< _Ch_type, _St, _Sa > & __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format string.

**Parameters**

<code>__s</code>	[IN] The C-string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

**Returns**

The string after replacing.

**Exceptions**

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [back\\_inserter\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_replace\(\)](#).

**regex\_replace()** [5/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
basic_string< _Ch_type, _St, _Sa > std::regex_replace (
    const basic_string< _Ch_type, _St, _Sa > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    const _Ch_type * __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format C-string.

**Parameters**

<code>__s</code>	[IN] The string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

**Returns**

The string after replacing.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [back\\_inserter\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::begin\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_replace\(\)](#).

**regex\_replace()** [6/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>
```

```
basic_string<_Ch_type, _St, _Sa > std::regex_replace (
    const basic_string<_Ch_type, _St, _Sa > & __s,
    const basic_regex<_Ch_type, _Rx_traits > & __e,
    const basic_string<_Ch_type, _Fst, _Fsa > & __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format string.

## Parameters

<code>__s</code>	[IN] The string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

## Returns

The string after replacing.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [back\\_inserter\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::begin\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_replace\(\)](#).

**regex\_search()** [1/7]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
```

```
bool std::regex_search (
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex<_Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Searches for a regular expression within a range.

## Parameters

<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__re</code>	[IN] The regular expression to search for.

<code>__flags</code>	[IN] Search policy flags.
----------------------	---------------------------

## Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

**regex\_search()** [2/7]

```
template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
    _Bi_iter __s,
    _Bi_iter __e,
    match_results< _Bi_iter, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Searches for a regular expression within a range.

## Parameters

<code>__s</code>	[IN] The start of the string to search.
<code>__e</code>	[IN] One-past-the-end of the string to search.
<code>__m</code>	[OUT] The match results.
<code>__re</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

## Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string, the content of <code>m</code> is undefined.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::regex\\_constants::match\\_default](#).

Referenced by [std::regex\\_iterator< const char \\* >::regex\\_iterator\(\)](#), [std::regex\\_iterator< \\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits >::operator++\(\)](#), [regex\\_search\(\)](#), [regex\\_search\(\)](#), [regex\\_search\(\)](#), [regex\\_search\(\)](#), and [regex\\_search\(\)](#).

**regex\_search()** [3/7]

```
template<typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
    const _Ch_type * __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default) [inline]
```

Searches for a regular expression within a C-string.

## Parameters

<code>__s</code>	[IN] The C-string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__f</code>	[IN] Search policy flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

References [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

**regex\_search()** [4/7]

```
template<typename _Ch_type, class _Alloc, class _Rx_traits>
bool std::regex_search (
    const _Ch_type * __s,
    match_results< const _Ch_type *, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default) [inline]
```

Searches for a regular expression within a C-string.

## Parameters

<code>__s</code>	[IN] A C-string to search for the regex.
<code>__m</code>	[OUT] The set of regex matches.
<code>__e</code>	[IN] The regex to search for in <code>s</code> .
<code>__f</code>	[IN] The search flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string, the content of <code>m</code> is undefined.



## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

**regex\_search()** [5/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↵
Rx_traits>
bool std::regex_search (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↵
iterator, _Alloc > & ,
    const basic_regex< _Ch_type, _Rx_traits > & ,
    regex_constants::match_flag_type = regex_constants::match_default) [delete]
```

Prevent unsafe attempts to get `match_results` from a temporary string.

References [std::regex\\_constants::match\\_default](#).

**regex\_search()** [6/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↵
Rx_traits>
bool std::regex_search (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↵
iterator, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default) [inline]
```

Searches for a regular expression within a string.

## Parameters

<code>_↵ _s</code>	[IN] A C++ string to search for the regex.
<code>_↵ _m</code>	[OUT] The set of regex matches.
<code>_↵ _e</code>	[IN] The regex to search for in <code>s</code> .
<code>_↵ _f</code>	[IN] The search flags.

## Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string, the content of <code>m</code> is undefined.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::begin\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

**regex\_search()** [7/7]

```
template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
    const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Searches for a regular expression within a string.

**Parameters**

<code>__s</code>	[IN] The string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

**Return values**

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

**Exceptions**

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

References [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

**swap()** [1/2]

```
template<typename _Ch_type, typename _Rx_traits>
void swap (
    basic_regex< _Ch_type, _Rx_traits > & __lhs,
    basic_regex< _Ch_type, _Rx_traits > & __rhs) [related]
```

Swaps the contents of two regular expression objects.

**Parameters**

<code>__lhs</code>	First regular expression.
<code>__rhs</code>	Second regular expression.

**swap()** [2/2]

```
template<typename _Bi_iter, typename _Alloc>
void swap (
    match_results< _Bi_iter, _Alloc > & __lhs,
    match_results< _Bi_iter, _Alloc > & __rhs) [related]
```

Swaps two match results.

**Parameters**

<code>__lhs</code>	A match result.
<code>__rhs</code>	A match result.

The contents of the two `match_results` objects are swapped.

### 3.12.4 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



#### Classes

- struct `std::__detail::_BracketMatcher<_TraitsT, __icase, __collate >`
- class `std::__detail::_Compiler<_TraitsT >`
- class `std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode >`
- class `std::__detail::_Scanner<_CharT >`
- class `std::__detail::_StateSeq<_TraitsT >`

#### Typedefs

- template<typename `_CharT`>  
using `std::__detail::Matcher`
- typedef long `std::__detail::StatelIdT`

#### Enumerations

- enum `std::__detail::_Opcode` : int {  
`_S_opcode_unknown` , `_S_opcode_alternative` , `_S_opcode_repeat` , `_S_opcode_backref` ,  
`_S_opcode_line_begin_assertion` , `_S_opcode_line_end_assertion` , `_S_opcode_word_boundary` , `_S_opcode_subexpr_lookahead` ,  
`_S_opcode_subexpr_begin` , `_S_opcode_subexpr_end` , `_S_opcode_dummy` , `_S_opcode_match` ,  
`_S_opcode_accept` }

#### Variables

- constexpr `_StatelIdT std::__detail::_S_invalid_state_id`

#### 3.12.4.1 Detailed Description

#### 3.12.4.2 Enumeration Type Documentation

##### `_Opcode`

```
enum std::__detail::_Opcode : int
```

Operation codes that define the type of transitions within the base NFA that represents the regular expression.

### 3.13 Strings

Collaboration diagram for Strings:



#### Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
- class `std::basic_string_view<_CharT, _Traits>`
- class `std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits>`
- struct `std::char_traits<_CharT>`

#### Typedefs

- typedef `basic_string<char>` `std::string`
- typedef `basic_string<char16_t>` `std::u16string`
- typedef `basic_string<char32_t>` `std::u32string`
- typedef `basic_string<wchar_t>` `std::wstring`

#### 3.13.1 Detailed Description

#### 3.13.2 Typedef Documentation

##### string

```
typedef basic_string<char> std::string
```

A string of `char`.

##### u16string

```
typedef basic_string<char16_t> std::u16string
```

A string of `char16_t`.

##### u32string

```
typedef basic_string<char32_t> std::u32string
```

A string of `char32_t`.

## wstring

```
typedef basic\_string<wchar_t> std::wstring  
A string of wchar_t.
```

### 3.14 Technical Specifications

Collaboration diagram for Technical Specifications:



#### Topics

- [Filesystem TS](#)
- [Library Fundamentals TS](#)
- [Parallelism TS](#)

#### 3.14.1 Detailed Description

Components specified by various Technical Specifications.

As indicated by the `std::experimental` namespace and the header paths, the contents of these Technical Specifications are experimental and not part of the C++ standard. As such the interfaces and implementations may change in the future, and there is **no guarantee of compatibility between different GCC releases** for these features.

#### 3.14.2 Filesystem TS

Collaboration diagram for Filesystem TS:



#### Files

- file [filesystem](#)

## Classes

- class `std::experimental::filesystem::v1::filesystem_error`
- class `std::experimental::filesystem::v1::path::iterator`
- class `std::experimental::filesystem::v1::path`
- struct `std::experimental::filesystem::v1::space_info`

## Typedefs

- using `std::experimental::filesystem::file_time_type`

## Enumerations

- enum class `std::experimental::filesystem::copy_options` : unsigned short { **none** , **skip\_existing** , **overwrite\_existing** , **update\_existing** , **recursive** , **copy\_symlinks** , **skip\_symlinks** , **directories\_only** , **create\_symlinks** , **create\_hard\_links** }
- enum class `std::experimental::filesystem::directory_options` : unsigned char { **none** , **follow\_directory\_symlink** , **skip\_permission\_denied** }
- enum class `std::experimental::filesystem::file_type` : signed char { **none** , **not\_found** , **regular** , **directory** , **symlink** , **block** , **character** , **fifo** , **socket** , **unknown** }
- enum class `std::experimental::filesystem::perms` : unsigned { **none** , **owner\_read** , **owner\_write** , **owner\_exec** , **owner\_all** , **group\_read** , **group\_write** , **group\_exec** , **group\_all** , **others\_read** , **others\_write** , **others\_exec** , **others\_all** , **all** , **set\_uid** , **set\_gid** , **sticky\_bit** , **mask** , **unknown** , **add\_perms** , **remove\_perms** , **symlink\_nofollow** }

## Functions

- `std::experimental::filesystem::v1::path::path` (const `path` &\_\_p)
- `std::experimental::filesystem::v1::path::path` (`path` &&\_\_p) noexcept
- `std::experimental::filesystem::v1::path::path` (`string_type` &&\_\_source)
- `path` `std::experimental::filesystem::absolute` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path` & `std::experimental::filesystem::v1::path::assign` (`string_type` &&\_\_source)
- `directory_iterator` `std::experimental::filesystem::begin` (`directory_iterator` \_\_iter) noexcept
- `recursive_directory_iterator` `std::experimental::filesystem::begin` (`recursive_directory_iterator` \_\_iter) noexcept
- `iterator` `std::experimental::filesystem::v1::path::begin` () const noexcept
- `path` `std::experimental::filesystem::canonical` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
- `path` `std::experimental::filesystem::canonical` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path` `std::experimental::filesystem::canonical` (const `path` &\_\_p, `error_code` &\_\_ec)
- int `std::experimental::filesystem::v1::path::compare` (const `basic_string_view`< value\_type > \_\_s) const
- int `std::experimental::filesystem::v1::path::compare` (const `string_type` &\_\_s) const
- int `std::experimental::filesystem::v1::path::compare` (const value\_type \*\_\_s) const
- void `std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to)
- void `std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options)
- void `std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options, `error_code` &) noexcept
- void `std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec) noexcept
- bool `std::experimental::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to)

- bool **std::experimental::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_option)
- bool **std::experimental::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_option, [error\\_code](#) &\_\_ec)
- bool **std::experimental::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, [error\\_code](#) &\_\_ec)
- void **std::experimental::filesystem::copy\_symlink** (const [path](#) &\_\_existing\_symlink, const [path](#) &\_\_new\_↵symlink)
- void **std::experimental::filesystem::copy\_symlink** (const [path](#) &\_\_existing\_symlink, const [path](#) &\_\_new\_↵symlink, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::create\_directories** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::create\_directories** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- bool **std::experimental::filesystem::create\_directory** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::create\_directory** (const [path](#) &\_\_p, const [path](#) &\_\_attributes)
- bool **std::experimental::filesystem::create\_directory** (const [path](#) &\_\_p, const [path](#) &\_\_attributes, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::create\_directory** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- void **std::experimental::filesystem::create\_directory\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void **std::experimental::filesystem::create\_directory\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink, [error\\_code](#) &\_\_ec) noexcept
- void **std::experimental::filesystem::create\_hard\_link** (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link)
- void **std::experimental::filesystem::create\_hard\_link** (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link, [error\\_code](#) &\_\_ec) noexcept
- void **std::experimental::filesystem::create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void **std::experimental::filesystem::create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **std::experimental::filesystem::current\_path** ()
- void **std::experimental::filesystem::current\_path** (const [path](#) &\_\_p)
- void **std::experimental::filesystem::current\_path** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **std::experimental::filesystem::current\_path** ([error\\_code](#) &\_\_ec)
- directory\_iterator **std::experimental::filesystem::end** (directory\_iterator) noexcept
- recursive\_directory\_iterator **std::experimental::filesystem::end** (recursive\_directory\_iterator) noexcept
- [iterator](#) **std::experimental::filesystem::v1::path::end** () const noexcept
- bool **std::experimental::filesystem::equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2)
- bool **std::experimental::filesystem::equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::exists** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::exists** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::exists** (file\_status \_\_s) noexcept
- [path](#) **std::experimental::filesystem::v1::path::extension** () const
- uintmax\_t **std::experimental::filesystem::file\_size** (const [path](#) &\_\_p)
- uintmax\_t **std::experimental::filesystem::file\_size** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **std::experimental::filesystem::v1::path::filename** () const
- [string](#) **std::experimental::filesystem::v1::path::generic\_string** () const
- template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>  
[std::basic\\_string](#)<\_CharT, \_Traits, \_Allocator> **std::experimental::filesystem::v1::path::generic\_string**  
(const \_Allocator &\_\_a= \_Allocator()) const
- [std::u16string](#) **std::experimental::filesystem::v1::path::generic\_u16string** () const
- [std::u32string](#) **std::experimental::filesystem::v1::path::generic\_u32string** () const
- [std::string](#) **std::experimental::filesystem::v1::path::generic\_u8string** () const
- [std::wstring](#) **std::experimental::filesystem::v1::path::generic\_wstring** () const
- uintmax\_t **std::experimental::filesystem::hard\_link\_count** (const [path](#) &\_\_p)
- uintmax\_t **std::experimental::filesystem::hard\_link\_count** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept

- `bool std::experimental::filesystem::v1::path::has_extension () const`
- `bool std::experimental::filesystem::v1::path::has_stem () const`
- `bool std::experimental::filesystem::v1::path::is_absolute () const`
- `bool std::experimental::filesystem::is_block_file (const path &__p)`
- `bool std::experimental::filesystem::is_block_file (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_block_file (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_character_file (const path &__p)`
- `bool std::experimental::filesystem::is_character_file (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_character_file (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_directory (const path &__p)`
- `bool std::experimental::filesystem::is_directory (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_directory (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_empty (const path &__p)`
- `bool std::experimental::filesystem::is_empty (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_fifo (const path &__p)`
- `bool std::experimental::filesystem::is_fifo (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_fifo (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_other (const path &__p)`
- `bool std::experimental::filesystem::is_other (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_other (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_regular_file (const path &__p)`
- `bool std::experimental::filesystem::is_regular_file (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_regular_file (file_status) noexcept`
- `bool std::experimental::filesystem::is_socket (const path &__p)`
- `bool std::experimental::filesystem::is_socket (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_socket (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_symlink (const path &__p)`
- `bool std::experimental::filesystem::is_symlink (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_symlink (file_status) noexcept`
- `file_time_type std::experimental::filesystem::last_write_time (const path &__p)`
- `file_time_type std::experimental::filesystem::last_write_time (const path &__p, error_code &__ec) noexcept`
- `void std::experimental::filesystem::last_write_time (const path &__p, file_time_type __new_time)`
- `void std::experimental::filesystem::last_write_time (const path &__p, file_time_type __new_time, error_code &__ec) noexcept`
- `path & std::experimental::filesystem::v1::path::make_preferred ()`
- `bool std::experimental::filesystem::operator!= (const directory_iterator &__lhs, const directory_iterator &__rhs)`
- `bool std::experimental::filesystem::operator!= (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)`
- `copy_options & std::experimental::filesystem::operator+= (copy_options &__x, copy_options __y) noexcept`
- `reference std::experimental::filesystem::v1::path::iterator::operator* () const noexcept`
- `iterator & std::experimental::filesystem::v1::path::iterator::operator++ () noexcept`
- `template<typename _CharT>  
__detail::_Path< _CharT *, _CharT * > & std::experimental::filesystem::v1::path::operator+= (_CharT __x)`
- `path & std::experimental::filesystem::v1::path::operator+= (basic_string_view< value_type > __x)`
- `path & std::experimental::filesystem::v1::path::operator+= (const path &__x)`
- `path & std::experimental::filesystem::v1::path::operator+= (const string_type &__x)`
- `path & std::experimental::filesystem::v1::path::operator+= (const value_type * __x)`
- `path & std::experimental::filesystem::v1::path::operator+= (value_type __x)`
- `iterator & std::experimental::filesystem::v1::path::iterator::operator-- () noexcept`
- `bool std::experimental::filesystem::operator< (const path &__lhs, const path &__rhs) noexcept`



- `path & std::experimental::filesystem::v1::path::operator=` (const `path` &\_\_p)
- `path & std::experimental::filesystem::v1::path::operator=` (`path` &&\_\_p) noexcept
- `path & std::experimental::filesystem::v1::path::operator=` (`string_type` &&\_\_source)
- `bool std::experimental::filesystem::operator==` (const `directory_iterator` &\_\_lhs, const `directory_iterator` &\_\_rhs)
- `bool std::experimental::filesystem::operator==` (const `path` &\_\_lhs, const `path` &\_\_rhs) noexcept
- `bool std::experimental::filesystem::operator==` (const `recursive_directory_iterator` &\_\_lhs, const `recursive_directory_iterator` &\_\_rhs)
- `constexpr copy_options std::experimental::filesystem::operator^` (`copy_options` \_\_x, `copy_options` \_\_y) noexcept
- `copy_options & std::experimental::filesystem::operator^=` (`copy_options` &\_\_x, `copy_options` \_\_y) noexcept
- `constexpr copy_options std::experimental::filesystem::operator|` (`copy_options` \_\_x, `copy_options` \_\_y) noexcept
- `copy_options & std::experimental::filesystem::operator|=` (`copy_options` &\_\_x, `copy_options` \_\_y) noexcept
- `constexpr copy_options std::experimental::filesystem::operator~` (`copy_options` \_\_x) noexcept
- `void std::experimental::filesystem::permissions` (const `path` &\_\_p, `perms` \_\_prms)
- `void std::experimental::filesystem::permissions` (const `path` &\_\_p, `perms` \_\_prms, `error_code` &\_\_ec) noexcept
- `path std::experimental::filesystem::read_symlink` (const `path` &\_\_p)
- `path std::experimental::filesystem::read_symlink` (const `path` &\_\_p, `error_code` &\_\_ec)
- `bool std::experimental::filesystem::remove` (const `path` &\_\_p)
- `bool std::experimental::filesystem::remove` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `uintmax_t std::experimental::filesystem::remove_all` (const `path` &\_\_p)
- `uintmax_t std::experimental::filesystem::remove_all` (const `path` &\_\_p, `error_code` &\_\_ec)
- `void std::experimental::filesystem::rename` (const `path` &\_\_from, const `path` &\_\_to)
- `void std::experimental::filesystem::rename` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec) noexcept
- `void std::experimental::filesystem::resize_file` (const `path` &\_\_p, `uintmax_t` \_\_size)
- `void std::experimental::filesystem::resize_file` (const `path` &\_\_p, `uintmax_t` \_\_size, `error_code` &\_\_ec) noexcept
- `space_info std::experimental::filesystem::space` (const `path` &\_\_p)
- `space_info std::experimental::filesystem::space` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `file_status std::experimental::filesystem::status` (const `path` &)
- `file_status std::experimental::filesystem::status` (const `path` &, `error_code` &) noexcept
- `bool std::experimental::filesystem::status_known` (`file_status`) noexcept
- `path std::experimental::filesystem::v1::path::stem` () const
- `std::string std::experimental::filesystem::v1::path::string` () const
- `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> std::basic_string<_CharT, _Traits, _Allocator> std::experimental::filesystem::v1::path::string` (const \_\_a= \_\_Allocator()) const
- `void std::experimental::filesystem::v1::path::swap` (`path` &\_\_rhs) noexcept
- `file_status std::experimental::filesystem::symlink_status` (const `path` &)
- `file_status std::experimental::filesystem::symlink_status` (const `path` &, `error_code` &) noexcept
- `path std::experimental::filesystem::system_complete` (const `path` &\_\_p)
- `path std::experimental::filesystem::system_complete` (const `path` &\_\_p, `error_code` &\_\_ec)
- `path std::experimental::filesystem::temp_directory_path` ()
- `path std::experimental::filesystem::temp_directory_path` (`error_code` &\_\_ec)
- `std::u16string std::experimental::filesystem::v1::path::u16string` () const
- `std::u32string std::experimental::filesystem::v1::path::u32string` () const
- `std::string std::experimental::filesystem::v1::path::u8string` () const
- `std::wstring std::experimental::filesystem::v1::path::wstring` () const

- constexpr perms **std::experimental::filesystem::operator|** (perms \_\_x, perms \_\_y) noexcept
- constexpr perms **std::experimental::filesystem::operator^** (perms \_\_x, perms \_\_y) noexcept
- constexpr perms **std::experimental::filesystem::operator~** (perms \_\_x) noexcept
- perms & **std::experimental::filesystem::operator&=** (perms &\_\_x, perms \_\_y) noexcept
- perms & **std::experimental::filesystem::operator|=** (perms &\_\_x, perms \_\_y) noexcept
- perms & **std::experimental::filesystem::operator^=** (perms &\_\_x, perms \_\_y) noexcept
- constexpr directory\_options **std::experimental::filesystem::operator|** (directory\_options \_\_x, directory\_options \_\_y) noexcept
- constexpr directory\_options **std::experimental::filesystem::operator^** (directory\_options \_\_x, directory\_options \_\_y) noexcept
- constexpr directory\_options **std::experimental::filesystem::operator~** (directory\_options \_\_x) noexcept
- directory\_options & **std::experimental::filesystem::operator&=** (directory\_options &\_\_x, directory\_options \_\_y) noexcept
- directory\_options & **std::experimental::filesystem::operator|=** (directory\_options &\_\_x, directory\_options \_\_y) noexcept
- directory\_options & **std::experimental::filesystem::operator^=** (directory\_options &\_\_x, directory\_options \_\_y) noexcept

### 3.14.2.1 Detailed Description

Utilities for performing operations on file systems and their components, such as paths, regular files, and directories.  
ISO/IEC TS 18822:2015 C++ File System Technical Specification

Since

C++11

Remarks

Link using `-lstdc++fs` to use these types and functions.

### 3.14.2.2 Typedef Documentation

#### file\_time\_type

using `std::experimental::filesystem::v1::file_time_type`

The type used for file timestamps.

### 3.14.2.3 Enumeration Type Documentation

#### copy\_options

enum class `std::experimental::filesystem::v1::copy_options` : unsigned short [strong]

Bitmask type controlling effects of `filesystem::copy`

#### directory\_options

enum class `std::experimental::filesystem::v1::directory_options` : unsigned char [strong]

Bitmask type controlling directory iteration.

#### file\_type

enum class `std::experimental::filesystem::v1::file_type` : signed char [strong]

Enumerated type representing the type of a file.

## perms

```
enum class std::experimental::filesystem::v1::perms : unsigned [strong]
```

Bitmask type representing file access permissions.

### 3.14.3 Library Fundamentals TS

Collaboration diagram for Library Fundamentals TS:



## Topics

- [Array creation functions](#)
- [Const-propagating wrapper](#)
- [Detection idiom](#)
- [Logical operator traits](#)
- [Optional values](#)
- [Type-safe container of any type](#)
- [Variable template for type traits](#)

## Files

- file [algorithm](#)
- file [any](#)
- file [array](#)
- file [chrono](#)
- file [deque](#)
- file [forward\\_list](#)
- file [functional](#)
- file [iterator](#)
- file [list](#)

- file [map](#)
- file [memory](#)
- file [memory\\_resource](#)
- file [numeric](#)
- file [optional](#)
- file [propagate\\_const](#)
- file [random](#)
- file [ratio](#)
- file [regex](#)
- file [set](#)
- file [string](#)
- file [string\\_view](#)
- file [system\\_error](#)
- file [tuple](#)
- file [type\\_traits](#)
- file [unordered\\_map](#)
- file [unordered\\_set](#)
- file [utility](#)
- file [vector](#)

## Classes

- class [std::experimental::fundamentals\\_v1::basic\\_string\\_view<\\_CharT, \\_Traits>](#)

### 3.14.3.1 Detailed Description

Components defined by the *C++ Extensions for Library Fundamentals* Technical Specification, versions 1 and 2.

- ISO/IEC TS 19568:2015 C++ Extensions for Library Fundamentals
- ISO/IEC TS 19568:2017 C++ Extensions for Library Fundamentals, Version 2

### 3.14.3.2 Array creation functions

Collaboration diagram for Array creation functions:



## Functions

- `template<typename _Tp, size_t _Nm, size_t... _Idx>  
constexpr array< remove\_cv\_t< _Tp >, _Nm > std::experimental::\_\_to\_array (_Tp(&__a)[_Nm],  
index\_sequence< _Idx... >)`
- `template<typename _Dest = void, typename... _Types>  
constexpr array< typename \_\_make\_array\_elem<\_Dest, \_Types...>::type, sizeof...\(\_Types\)> std::experimental::make\\_array  
\(\_Types &&... \_\_t\)`

- `template<typename _Tp, size_t _Nm>`  
`constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::to\_array (_Tp(&__a)[_Nm]) noexcept(is_nothrow_constructible<`  
`remove_cv_t< _Tp >, _Tp & >::value)`

### 3.14.3.2.1 Detailed Description

Array creation functions as described in N4529, Working Draft, C++ Extensions for Library Fundamentals, Version 2

### 3.14.3.2.2 Function Documentation

#### `make_array()`

```
template<typename _Dest = void, typename... _Types>
array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::fundamentals\_v2::make\_array (
    _Types &&... __t) [constexpr]
```

Create a `std::array` from a variable-length list of arguments.

#### `to_array()`

```
template<typename _Tp, size_t _Nm>
array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals\_v2::to\_array (
    _Tp(&) __a[_Nm]) [constexpr], [noexcept]
```

Create a `std::array` from an array.

### 3.14.3.3 Const-propagating wrapper

Collaboration diagram for Const-propagating wrapper:



## Classes

- class `std::experimental::fundamentals\_v2::propagate\_const< _Tp >`

## Functions

- `template<typename _Tp>`  
`constexpr const _Tp & std::experimental::get\_underlying (const propagate\_const< _Tp > &__pt) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp & std::experimental::get\_underlying (propagate\_const< _Tp > &__pt) noexcept`
- `constexpr std::experimental::fundamentals\_v2::\_\_propagate\_const\_conversion\_nc< _Tp, _Elem, true >::operator _Elem * ()`
- `constexpr std::experimental::fundamentals\_v2::\_\_propagate\_const\_conversions< _Tp * >::operator _Tp * () noexcept`
- `constexpr std::experimental::fundamentals\_v2::\_\_propagate\_const\_conversion\_c< _Tp, _Elem, true >::operator const _Elem * () const`

- constexpr **std::experimental::fundamentals\_v2::\_\_propagate\_const\_conversions**< \_Tp \* >::operator **const** \_Tp \* () const noexcept
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator!=** (const \_Tp &\_\_t, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator!=** (const [propagate\\_const](#)< \_Tp > &\_\_pt, const \_Up &\_\_u)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator!=** (const [propagate\\_const](#)< \_Tp > &\_\_pt, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp>  
constexpr bool **std::experimental::operator!=** (const [propagate\\_const](#)< \_Tp > &\_\_pt, nullptr\_t)
- template<typename \_Tp>  
constexpr bool **std::experimental::operator!=** (nullptr\_t, const [propagate\\_const](#)< \_Tp > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**< (const \_Tp &\_\_t, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**< (const [propagate\\_const](#)< \_Tp > &\_\_pt, const \_Up &\_\_u)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**< (const [propagate\\_const](#)< \_Tp > &\_\_pt, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**<= (const \_Tp &\_\_t, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**<= (const [propagate\\_const](#)< \_Tp > &\_\_pt, const \_Up &\_\_u)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**<= (const [propagate\\_const](#)< \_Tp > &\_\_pt, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**== (const \_Tp &\_\_t, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**== (const [propagate\\_const](#)< \_Tp > &\_\_pt, const \_Up &\_\_u)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**== (const [propagate\\_const](#)< \_Tp > &\_\_pt, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp>  
constexpr bool **std::experimental::operator**== (const [propagate\\_const](#)< \_Tp > &\_\_pt, nullptr\_t)
- template<typename \_Tp>  
constexpr bool **std::experimental::operator**== (nullptr\_t, const [propagate\\_const](#)< \_Tp > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**> (const \_Tp &\_\_t, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**> (const [propagate\\_const](#)< \_Tp > &\_\_pt, const \_Up &\_\_u)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**> (const [propagate\\_const](#)< \_Tp > &\_\_pt, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**>= (const \_Tp &\_\_t, const [propagate\\_const](#)< \_Up > &\_\_pu)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**>= (const [propagate\\_const](#)< \_Tp > &\_\_pt, const \_Up &\_\_u)
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::experimental::operator**>= (const [propagate\\_const](#)< \_Tp > &\_\_pt, const [propagate\\_const](#)< \_Up > &\_\_pu)

- `template<typename _Tp>`  
`constexpr enable\_if\_t< __is_swappable< _Tp >::value, void > std::experimental::swap (propagate\_const<`  
`_Tp > &__pt, propagate\_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)`

#### 3.14.3.3.1 Detailed Description

A const-propagating wrapper that propagates const to pointer-like members, as described in n4388 "A Proposal to Add a Const-Propagating Wrapper to the Standard Library".

#### 3.14.3.4 Detection idiom

Collaboration diagram for Detection idiom:



- `#define \_\_cpp\_lib\_experimental\_detect`
- `template<typename...>`  
`using std::experimental::void\_t`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::detected\_or`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::detected\_or\_t`
- `template<template< typename... > class _Op, typename... _Args>`  
`using std::experimental::detected\_t`
- `template<template< typename... > class _Op, typename... _Args>`  
`using std::experimental::is\_detected`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::is\_detected\_exact`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::is\_detected\_convertible`
- `template<template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::is\_detected\_v`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::is\_detected\_exact\_v`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::is\_detected\_convertible\_v`

#### 3.14.3.4.1 Detailed Description

Since

Library Fundamentals TS v2. C++14.

#### 3.14.3.4.2 Macro Definition Documentation

##### **`__cpp_lib_experimental_detect`**

```
#define __cpp_lib_experimental_detect
```

A metafunction that always yields void, used for detecting valid types.

#### 3.14.3.4.3 Typedef Documentation

##### **`detected_or`**

```
template<typename _Default, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::detected_or
```

A metafunction that always yields void, used for detecting valid types.

##### **`detected_or_t`**

```
template<typename _Default, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::detected_or_t
```

A metafunction that always yields void, used for detecting valid types.

##### **`detected_t`**

```
template<template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::detected_t
```

A metafunction that always yields void, used for detecting valid types.

##### **`is_detected`**

```
template<template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected
```

A metafunction that always yields void, used for detecting valid types.

##### **`is_detected_convertible`**

```
template<typename _To, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected_convertible
```

A metafunction that always yields void, used for detecting valid types.

##### **`is_detected_exact`**

```
template<typename _Expected, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected_exact
```

A metafunction that always yields void, used for detecting valid types.

##### **`void_t`**

```
template<typename...>
using std::experimental::fundamentals_v2::void_t
```

A metafunction that always yields void, used for detecting valid types.

#### 3.14.3.4.4 Variable Documentation

##### **`is_detected_convertible_v`**

```
template<typename _To, template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_convertible_v [constexpr]
```



A metafunction that always yields void, used for detecting valid types.

### is\_detected\_exact\_v

```
template<typename _Expected, template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_exact_v [constexpr]
```

A metafunction that always yields void, used for detecting valid types.

### is\_detected\_v

```
template<template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_v [constexpr]
```

A metafunction that always yields void, used for detecting valid types.

## 3.14.3.5 Logical operator traits

Collaboration diagram for Logical operator traits:



- `#define __cpp_lib_experimental_logical_traits`
- `template<typename... _Bn>`  
`constexpr bool std::experimental::conjunction_v`
- `template<typename... _Bn>`  
`constexpr bool std::experimental::disjunction_v`
- `template<typename _Pp>`  
`constexpr bool std::experimental::negation_v`

### 3.14.3.5.1 Detailed Description

Since

Library Fundamentals TS v2. C++14.

### 3.14.3.6 Optional values

Collaboration diagram for Optional values:



**Classes**

- class `std::experimental::fundamentals_v1::bad_optional_access`
- struct `std::experimental::fundamentals_v1::in_place_t`
- struct `std::experimental::fundamentals_v1::nullopt_t`
- class `std::experimental::fundamentals_v1::optional<_Tp>`

**Macros**

- `#define __cpp_lib_experimental_optional`

**Variables**

- constexpr `in_place_t` `std::experimental::in_place`
- constexpr `nullopt_t` `std::experimental::nullopt`

**3.14.3.6.1 Detailed Description**

Class template for optional values and surrounding facilities, as described in n3793 "A proposal to add a utility class to represent optional objects (Revision 5)".

**3.14.3.6.2 Variable Documentation****in\_place**

`in_place_t` `std::experimental::fundamentals_v1::in_place` [constexpr]

Tag for in-place construction.

**nullopt**

`nullopt_t` `std::experimental::fundamentals_v1::nullopt` [constexpr]

Tag to disengage optional objects.

**3.14.3.7 Type-safe container of any type**

Collaboration diagram for Type-safe container of any type:

**Classes**

- class `std::experimental::fundamentals_v1::any`
- class `std::experimental::fundamentals_v1::bad_any_cast`

**Macros**

- `#define __cpp_lib_experimental_any`

## Functions

- static void **std::experimental::fundamentals\_v1::any::\_Manager\_external**< \_Tp >::\_S\_manage ( \_Op \_\_↵ which, const [any](#) \* \_\_anyp, \_Arg \* \_\_arg)
- static void **std::experimental::fundamentals\_v1::any::\_Manager\_internal**< \_Tp >::\_S\_manage ( \_Op \_\_↵ which, const [any](#) \* \_\_anyp, \_Arg \* \_\_arg)
- template<typename \_ValueType>  
\_ValueType **std::experimental::any\_cast** (const [any](#) & \_\_any)
- void **std::experimental::swap** ([any](#) & \_\_x, [any](#) & \_\_y) noexcept
- template<typename \_ValueType>  
\_ValueType **std::experimental::any\_cast** ([any](#) & \_\_any)
- template<typename \_ValueType, typename [enable\\_if](#)<!is\_move\_constructible< \_ValueType >::value||is\_lvalue\_reference< \_ValueType >::value, bool >::type = true>  
\_ValueType **std::experimental::any\_cast** ([any](#) && \_\_any)
- template<typename \_ValueType>  
const \_ValueType \* **std::experimental::any\_cast** (const [any](#) \* \_\_any) noexcept
- template<typename \_ValueType>  
\_ValueType \* **std::experimental::any\_cast** ([any](#) \* \_\_any) noexcept

### 3.14.3.7.1 Detailed Description

A type-safe container for single values of value types, as described in n3804 "Any Library Proposal (Revision 3)".

### 3.14.3.7.2 Function Documentation

#### **any\_cast()** [1/5]

```
template<typename _ValueType, typename enable\_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>
_ValueType std::experimental::fundamentals_v1::any_cast (
    any && __any) [inline]
```

Access the contained object.

#### Template Parameters

<code>_ValueType</code>	A reference or CopyConstructible type.
-------------------------	--

#### Parameters

<code>__any</code>	The object to access.
--------------------	-----------------------

#### Returns

The contained object.

#### Exceptions

<code>bad_any_cast</code>	If <code>__any.type() != typeid(remove_reference_t&lt;_ValueType&gt;)</code>
---------------------------	--

**any\_cast()** [2/5]

```
template<typename _ValueType>
_ValueType std::experimental::fundamentals_v1::any_cast (
    any & __any) [inline]
```

Access the contained object.

**Template Parameters**

<code>_ValueType</code>	A reference or CopyConstructible type.
-------------------------	--

**Parameters**

<code>__any</code>	The object to access.
--------------------	-----------------------

**Returns**

The contained object.

**Exceptions**

<code>bad_any_cast</code>	If <code>__any.type() != typeid(remove_reference_t&lt;_ValueType&gt;)</code>
---------------------------	--

**any\_cast()** [3/5]

```
template<typename _ValueType>
_ValueType * std::experimental::fundamentals_v1::any_cast (
    any * __any) [inline], [noexcept]
```

Access the contained object.

**Template Parameters**

<code>_ValueType</code>	The type of the contained object.
-------------------------	-----------------------------------

**Parameters**

<code>__any</code>	A pointer to the object to access.
--------------------	------------------------------------

**Returns**

The address of the contained object if `__any != nullptr && __any.type() == typeid(_ValueType)` , otherwise a null pointer.

**any\_cast()** [4/5]

```
template<typename _ValueType>
_ValueType std::experimental::fundamentals_v1::any_cast (
    const any & __any) [inline]
```

Access the contained object.

### Template Parameters

<code>_ValueType</code>	A const-reference or CopyConstructible type.
-------------------------	--

### Parameters

<code>__any</code>	The object to access.
--------------------	-----------------------

### Returns

The contained object.

### Exceptions

<code>bad_any_cast</code>	If <code>__any.type() != typeid(remove_reference_t&lt;_ValueType&gt;)</code>
---------------------------	--

### **any\_cast()** [5/5]

```
template<typename _ValueType>
const _ValueType * std::experimental::fundamentals_v1::any_cast (
    const any * __any) [inline], [noexcept]
```

Access the contained object.

### Template Parameters

<code>_ValueType</code>	The type of the contained object.
-------------------------	-----------------------------------

### Parameters

<code>__any</code>	A pointer to the object to access.
--------------------	------------------------------------

### Returns

The address of the contained object if `__any != nullptr && __any.type() == typeid(_ValueType)` , otherwise a null pointer.

### **swap()**

```
void std::experimental::fundamentals_v1::swap (
    any & __x,
    any & __y) [inline], [noexcept]
```

Exchange the states of two any objects.

### 3.14.3.8 Variable template for type traits

Collaboration diagram for Variable template for type traits:



- `#define __cpp_lib_experimental_type_trait_variable_templates`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_void_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_null_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_integral_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_floating_point_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_array_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_lvalue_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_rvalue_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_object_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_function_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_enum_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_union_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_class_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_function_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_arithmetic_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_fundamental_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_object_v`

- `template<typename _Tp>`  
`constexpr bool std::experimental::is_scalar_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_compound_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_const_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_volatile_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivial_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copyable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_standard_layout_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_pod_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_literal_type_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_empty_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_polymorphic_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_abstract_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_final_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_signed_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_unsigned_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_destructible_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_trivially_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_default_constructible_v`

- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_trivially_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_destructible_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_nothrow_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_nothrow_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_destructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::has_virtual_destructor_v`
- `template<typename _Tp>`  
`constexpr size_t std::experimental::alignment_of_v`
- `template<typename _Tp>`  
`constexpr size_t std::experimental::rank_v`
- `template<typename _Tp, unsigned _Idx = 0>`  
`constexpr size_t std::experimental::extent_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_same_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_same_v< _Tp, _Tp >`
- `template<typename _Base, typename _Derived>`  
`constexpr bool std::experimental::is_base_of_v`
- `template<typename _From, typename _To>`  
`constexpr bool std::experimental::is_convertible_v`

### 3.14.3.8.1 Detailed Description

Since

Library Fundamentals TS v1. C++14.

See also

`variable_templates`



### 3.14.4 Parallelism TS

Collaboration diagram for Parallelism TS:



#### Topics

- [Data parallel extensions](#)

#### 3.14.4.1 Detailed Description

Components defined by the *C++ Extensions for Parallelism* Technical Specification.

- ISO/IEC TS 19570:2015 C++ Extensions for Parallelism
- ISO/IEC TS 19570:2018 C++ Extensions for Parallelism, Version 2

#### 3.14.4.2 Data parallel extensions

Collaboration diagram for Data parallel extensions:



#### Macros

- `#define __cpp_lib_experimental_parallel_simd`

#### 3.14.4.2.1 Detailed Description

Data-parallel types library.

Since

C++17

### 3.15 Utilities

Collaboration diagram for Utilities:



#### Topics

- [Function Objects](#)
- [Memory](#)
- [Metaprogramming](#)
- [Rational Arithmetic](#)
- [Time](#)

#### Classes

- class [std::bitset<\\_Nb>](#)
- struct [std::pair<\\_T1, \\_T2>](#)
- struct [std::piecewise\\_construct\\_t](#)
- class [std::tuple<\\_Elements>](#)
- class [std::tuple<\\_T1, \\_T2>](#)
- struct [std::tuple\\_element<\\_\\_i, tuple<\\_Types...>>](#)
- struct [std::tuple\\_size<tuple<\\_Elements...>>](#)
- struct [std::type\\_index](#)
- struct [std::uses\\_allocator<tuple<\\_Types...>, \\_Alloc>](#)

#### Typedefs

- template<typename \_Tp, typename \_Up>  
using [std::\\_\\_like\\_t](#)

## Functions

- `template<typename... _Args1, typename... _Args2>`  
`constexpr std::pair<_T1, _T2>::pair (piecewise_construct_t, tuple<_Args1...>, tuple<_Args2...>)`
- `template<typename _Tp>`  
`constexpr _Tp * std::__addressof (_Tp &__r) noexcept`
- `template<typename _Tp, typename _Up = _Tp>`  
`constexpr _Tp std::__exchange (_Tp &__obj, _Up &&__new_val)`
- `template<size_t __i, typename _Head, typename... _Tail>`  
`constexpr _Head & std::__get_helper (_Tuple_impl< __i, _Head, _Tail...> &__t) noexcept`
- `template<size_t __i, typename _Head, typename... _Tail>`  
`constexpr const _Head & std::__get_helper (const _Tuple_impl< __i, _Head, _Tail...> &__t) noexcept`
- `template<size_t __i, typename... _Types>`  
`__enable_if_t<(__i >= sizeof...( _Types))> std::__get_helper (const tuple<_Types...> &)=delete`
- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>`  
`constexpr _Up && std::__invfwd (typename remove_reference<_Tp>::type &__t) noexcept`
- `template<typename _Callable, typename... _Args>`  
`constexpr __invoke_result<_Callable, _Args...>::type std::__invoke (_Callable &&__fn, _Args &&... __args)`  
`noexcept(__is_nothrow_invocable<_Callable, _Args...>::value)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemPtr, typename _Tp>`  
`constexpr _Res std::__invoke_impl (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _MemPtr, typename _Tp>`  
`constexpr _Res std::__invoke_impl (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _Fn, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_other, _Fn &&__f, _Args &&... __args)`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`constexpr enable_if_t<is_invocable_r_v<_Res, _Callable, _Args...>, _Res> std::__invoke_r (_Callable &&__fn, _Args &&... __args) noexcept(is_nothrow_invocable_r_v<_Res, _Callable, _Args...>)`
- `template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>`  
`constexpr _Cat std::__tuple_cmp (const _Tp &__t, const _Up &__u, _IndexSeq __indices)`
- `template<typename _Tp>`  
`constexpr _Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp>`  
`const _Tp * std::addressof (const _Tp &&)=delete`
- `template<typename _Tp>`  
`auto std::declval () noexcept -> decltype(__declval<_Tp>())`
- `template<typename _Tp>`  
`constexpr _Tp && std::forward (typename std::remove_reference<_Tp>::type &&__t) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp && std::forward (typename std::remove_reference<_Tp>::type &__t) noexcept`
- `template<typename... _Elements>`  
`constexpr tuple<_Elements &&...> std::forward_as_tuple (_Elements &&... __args) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple<_Elements...> > && std::get (const tuple<_Elements...> &&__t) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple<_Elements...> > & std::get (const tuple<_Elements...> &__t) noexcept`

- `template<size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > && std::get (tuple< _Elements... > &&__t) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > & std::get (tuple< _Elements... > &__t) noexcept`
- `template<typename _T1, typename _T2>`  
`constexpr pair< typename __decay_and_strip< _T1 >::type, typename __decay_and_strip< _T2 >::type > std::make_pair ( _T1 &&__x, _T2 &&__y)`
- `template<typename... _Elements>`  
`constexpr tuple< typename __decay_and_strip< _Elements >::type... > std::make_tuple ( _Elements &&... __args)`
- `template<typename _Tp>`  
`constexpr std::remove_reference< _Tp >::type && std::move ( _Tp &&__t) noexcept`
- `template<typename _Tp>`  
`constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_noexcept ( _Tp &__x) noexcept`
- `template<typename... _Tps, typename... _Ups>`  
`requires (sizeof...( _Tps) == sizeof...( _Ups)) && (requires { typename __detail::__synth3way_t< _Tps, _Ups>; } && ...)`  
`constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > std::operator<=> (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`
- `template<typename... _Tps, typename... _Ups>`  
`requires (sizeof...( _Tps) == sizeof...( _Ups)) && (requires (const _Tps& __t, const _Ups& __u) { { __t == __u } -> __detail::__boolean_testable; } && ...)`  
`constexpr bool std::operator== (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`
- `template<typename _Tp>`  
`requires (! __is_tuple_like< _Tp >::value) && is_move_constructible_v< _Tp> && is_move_assignable_v< _Tp>`  
`constexpr void std::swap ( _Tp &__a, _Tp &__b) noexcept(*conditional *) is_nothrow_move_assignable< _Tp > >`
- `template<typename _Tp, size_t _Nm>`  
`requires is_swappable_v< _Tp>`  
`constexpr void std::swap ( _Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(*conditional *)`
- `template<typename... _Elements>`  
`constexpr enable_if< __and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &... > std::tie ( _Elements &... __args) noexcept`
- `template<typename... _UTypes>`  
`std::tuple ( _UTypes... ) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename... _UTypes>`  
`std::tuple ( allocator_arg_t, _Alloc, _UTypes... ) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename _T1, typename _T2>`  
`std::tuple ( allocator_arg_t, _Alloc, pair< _T1, _T2 > ) -> tuple< _T1, _T2 >`
- `template<typename _Alloc, typename... _UTypes>`  
`std::tuple ( allocator_arg_t, _Alloc, tuple< _UTypes... > ) -> tuple< _UTypes... >`
- `template<typename _T1, typename _T2>`  
`std::tuple ( pair< _T1, _T2 > ) -> tuple< _T1, _T2 >`
- `template<typename... _Tpls, typename = typename enable_if< __and< __is_tuple_like< _Tpls >... >::value >::type>`  
`constexpr auto std::tuple_cat ( _Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::type`

## Variables

- `constexpr piecewise_construct_t std::piecewise_construct`

- `template<typename... _Types>`  
`constexpr size_t std::tuple_size_v< const tuple< _Types... > >`
- `template<typename... _Types>`  
`constexpr size_t std::tuple_size_v< tuple< _Types... > >`
- `template<typename _T1, typename _T2>`  
`pair ( _T1, _T2 ) -> pair< _T1, _T2 >`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)`
- `template<typename _T1, typename _T2>`  
`constexpr enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _T1, typename _T2>`  
`constexpr enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)`

### 3.15.1 Detailed Description

Basic function and class templates used with the rest of the library. Includes `pair`, `swap`, forward/move helpers, `declval`, `integer_sequence`.

### 3.15.2 Function Documentation

#### `pair()` [1/2]

```
template<typename _T1, typename _T2>
pair (
    _T1 ,
    _T2 ) -> pair< _T1, _T2 > [related]
```

Two pairs are equal iff their members are equal.

#### `pair()` [2/2]

```
template<class _T1, class _T2>
template<typename... _Args1, typename... _Args2>
std::pair< _T1, _T2 >::pair (
    piecewise_construct_t ,
    tuple< _Args1... > __first,
    tuple< _Args2... > __second) [inline], [constexpr]
```

"piecewise construction" using a tuple of arguments for each member.

#### Parameters

<code>__first</code>	Arguments for the first member of the pair.
<code>__second</code>	Arguments for the second member of the pair.

The elements of each tuple will be used as the constructor arguments for the data members of the pair.

**\_\_addressof()**

```
template<typename _Tp>
_Tp * std::__addressof (
    _Tp & __r) [inline], [constexpr], [noexcept]
```

Same as C++11 `std::addressof`.

Referenced by `std::back_insert_iterator< _Container >::back_insert_iterator()`, `std::front_insert_iterator< _Container >::front_insert_iterator()`, `std::insert_iterator< _Container >::insert_iterator()`, `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator()`, `std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator()`, `std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator()`, `__gnu_debug::__Safe_sequence< _Sequence >::M_transfer_from_if()`, `std::basic_ios< _CharT, _Traits >::copyfmt()`, `__gnu_debug::basic_string< char >::find_last_of()`, `std::forward_list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::merge()`, `std::sub_match< const char * >::operator<<()`, `__gnu_debug::__Safe_iterator< _Base_iterator, map >::operator=()`, `std::deque< _Tp, _Alloc >::operator=()`, `std::forward_list< _Tp, _Alloc >::operator=()`, `std::list< _Tp, _Alloc >::operator=()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::sub_match< const char * >::operator=()`, `std::vector< _Tp, polymorphic_allocator< _Tp > >< _BigBlock >::reserve()`, `std::list< _Tp, polymorphic_allocator< _Tp > >::splice()`, and `std::list< _Tp, polymorphic_allocator< _Tp > >::splice()`.

**\_\_invoke()**

```
template<typename _Callable, typename... _Args>
__invoke_result< _Callable, _Args... >::type std::__invoke (
    _Callable && __fn,
    _Args &&... __args) [constexpr], [noexcept]
```

Invoke a callable object.

References [forward\(\)](#).

**addressof()**

```
template<typename _Tp>
_Tp * std::addressof (
    _Tp & __r) [inline], [constexpr], [noexcept]
```

Returns the actual address of the object or function referenced by `r`, even in the presence of an overloaded operator`&`.

**Parameters**

←	Reference to an object or function.
←	
←	
←	
<i>r</i>	

**Returns**

The actual address.

**Since**

C++11

Referenced by `std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref()`, `std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref()`, `_Destroy()`, `_Destroy_n()`, `std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator=()`, and `std::move_only_function< _Res`

**declval()**

```
template<typename _Tp>
auto std::declval () -> decltype(__declval< _Tp >(0)) [noexcept]
```

Utility to simplify expressions used in unevaluated operands

Since

C++11

Referenced by [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >::vector\(\)](#), and [std::unique\\_ptr< \\_State >::operator\\*\(\)](#).

**forward()** [1/2]

```
template<typename _Tp>
_Tp && std::forward (
    typename std::remove_reference< _Tp >::type && __t) [constexpr], [noexcept]
```

Forward an rvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Since

C++11

**forward()** [2/2]

```
template<typename _Tp>
_Tp && std::forward (
    typename std::remove_reference< _Tp >::type & __t) [constexpr], [noexcept]
```

Forward an lvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Since

C++11

Referenced by [std::copyable\\_function< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::copyable\\_function\(\)](#), [std::copyable\\_function< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::copyable\\_function\(\)](#), [std::copyable\\_function< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::function\(\)](#), [std::move\\_only\\_function< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::move\\_only\\_function\(\)](#), [std::move\\_only\\_function< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::move\\_only\\_function\(\)](#), [std::move\\_only\\_function< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::move\\_only\\_function\(\)](#), [std::unique\\_ptr< \\_State >::unique\\_ptr\(\)](#), [\\_\\_invoke\(\)](#), [\\_Construct\(\)](#), [std::list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::\\_M\\_create\\_node\(\)](#), [std::deque< \\_Tp, \\_Alloc >::\\_M\\_push\\_back\\_aux\(\)](#), [std::deque< \\_Tp, \\_Alloc >::\\_M\\_push\\_front\\_aux\(\)](#), [std::shared\\_ptr< \\_State\\_base >::shared\\_ptr\(\)](#), [std::basic\\_string< \\_CharT >::append\(\)](#), [std::allocator\\_traits< \\_OuterAlloc >::construct\(\)](#), [std::allocator\\_traits< allocator< \\_Tp > >::construct\(\)](#), [std::allocator\\_traits< allocator< void > >::construct\(\)](#), [std::allocator\\_traits< pmr::polymorphic\\_allocator< \\_Tp > >::construct\(\)](#), [std::deque< \\_Tp, \\_Alloc >::emplace\(\)](#), [std::list< \\_Tp, \\_Alloc >::emplace\(\)](#), [std::map< \\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, const \\_Tp > > >::map\(\)](#), [std::multimap< \\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, const \\_Tp > > >::multimap\(\)](#), [std::multiset< \\_Key, \\_Cmp, polymorphic\\_allocator< \\_Key > >::multiset\(\)](#), [std::set< \\_Key, \\_Cmp, polymorphic\\_allocator< \\_Key > >::set\(\)](#), [std::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, polymorphic\\_allocator< pair< const \\_Key, const \\_Tp > > >::unordered\\_map\(\)](#), [std::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, polymorphic\\_allocator< pair< const \\_Key, const \\_Tp > > >::unordered\\_multimap\(\)](#), [std::unordered\\_multiset< \\_Key, \\_Hash, \\_Pred, polymorphic\\_allocator< \\_Key > >::unordered\\_multiset\(\)](#), [std::unordered\\_set< \\_Key, \\_Hash, \\_Pred, polymorphic\\_allocator< \\_Key > >::unordered\\_set\(\)](#).

```

std::vector< _Tp, polymorphic_allocator< _Tp > >::emplace(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_after(),
std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_front(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace_hint(),
std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace_hint(), std::multiset< _Key, _Cmp, polymorphic_allocator< _Key > >::emplace_hint(),
std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::emplace_hint(), std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace_hint(),
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace_hint(),
std::unordered_multiset< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::emplace_hint(), std::unordered_set< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::emplace_hint(),
get(), get(), get(), get(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign(),
make_pair(), std::shared_ptr< _State_base >::make_shared(), std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator(),
std::function< _Res(_ArgTypes...)>::operator(), std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator(),
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator(), std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator(),
std::function< _Res(_ArgTypes...)>::operator=(), std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator=(),
std::unique_ptr< _State >::operator=(), std::unique_ptr< _Tp[], _Dp >::operator=(), sample(), and throw_with_nested().

```

### forward\_as\_tuple()

```

template<typename... _Elements>
tuple< _Elements &&... > std::forward_as_tuple (
    _Elements &&... __args) [constexpr, [noexcept]]

```

Create a tuple of lvalue or rvalue references to the arguments.

Referenced by `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign()`,  
and `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign()`.

### get() [1/4]

```

template<size_t __i, typename... _Elements>
const __tuple_element_t< __i, tuple< _Elements... > > && std::get (
    const tuple< _Elements... > && __t) [constexpr, [noexcept]]

```

Return a const rvalue reference to the ith element of a const tuple rvalue.

### get() [2/4]

```

template<size_t __i, typename... _Elements>
const __tuple_element_t< __i, tuple< _Elements... > > & std::get (
    const tuple< _Elements... > & __t) [constexpr, [noexcept]]

```

Return a const reference to the ith element of a const tuple.

### get() [3/4]

```

template<size_t __i, typename... _Elements>
__tuple_element_t< __i, tuple< _Elements... > > && std::get (
    tuple< _Elements... > && __t) [constexpr, [noexcept]]

```

Return an rvalue reference to the ith element of a tuple rvalue.

### get() [4/4]

```

template<size_t __i, typename... _Elements>
__tuple_element_t< __i, tuple< _Elements... > > & std::get (
    tuple< _Elements... > & __t) [constexpr, [noexcept]]

```

Return a reference to the ith element of a tuple.



**make\_pair()**

```
template<typename _T1, typename _T2>
pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type >
std::make_pair (
    _T1 && __x,
    _T2 && __y) [constexpr]
```

A convenience wrapper for creating a pair from two objects.

**Parameters**

$\leftarrow$ _x	The first object.
$\leftarrow$ _y	The second object.

**Returns**

A newly-constructed pair<> object of the appropriate type.

The C++98 standard says the objects are passed by reference-to-const, but C++03 says they are passed by value (this was LWG issue #181).

Since C++11 they have been passed by forwarding reference and then forwarded to the new members of the pair. To create a pair with a member of reference type, pass a `reference_wrapper` to this function.

References [forward\(\)](#).

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), [\\_\\_gen\\_two\\_uniform\\_ints\(\)](#), [\\_\\_gnu\\_debug::\\_\\_get\\_distance\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_merge\\_advance\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\\_qsb\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector::\\_\\_M\\_sequential\\_algorithm\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector<\\_FIterator>::\\_\\_M\\_sequential\\_algorithm\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector::\\_\\_M\\_sequential\\_algorithm\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_multiway\\_merge\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#), and [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_\\_Node\\_cite](#)

**make\_tuple()**

```
template<typename... _Elements>
tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (
    _Elements &&... __args) [constexpr]
```

Create a tuple containing copies of the arguments.

**move()**

```
template<typename _Tp>
std::remove_reference< _Tp >::type && std::move (
    _Tp && __t) [constexpr], [noexcept]
```

Convert a value to an rvalue.

**Parameters**

$\leftarrow$	A thing of arbitrary type.
$\leftarrow$	
$\leftarrow$	
$\leftarrow$	
<i>t</i>	



### Returns

The parameter, possibly cast to an rvalue-reference.

Same as `std::move` unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

### Since

C++11

References [move\(\)](#).

### `operator<=>()` [1/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2,
_U2 > > operator<=> (
    const pair< _T1, _T2 > & __x,
    const pair< _U1, _U2 > & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

### `operator<=>()` [2/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2,
_U2 > > operator<=> (
    const pair< _T1, _T2 > & __x,
    const pair< _U1, _U2 > & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

### `operator==()` [1/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
bool operator== (
    const pair< _T1, _T2 > & __x,
    const pair< _U1, _U2 > & __y) [related]
```

Two pairs are equal iff their members are equal.

### `operator==()` [2/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
bool operator== (
    const pair< _T1, _T2 > & __x,
    const pair< _U1, _U2 > & __y) [related]
```

Two pairs are equal iff their members are equal.

### `swap()` [1/5]

```
template<typename _T1, typename _T2>
enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (
    pair< _T1, _T2 > & __x,
    pair< _T1, _T2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

**Note**

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

**swap() [2/5]**

```
template<typename _T1, typename _T2>
enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (
    pair< _T1, _T2 > & __x,
    pair< _T1, _T2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

**Note**

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

**swap() [3/5]**

```
template<typename _Tp>
requires (! __is_tuple_like<_Tp>::value) && is_move_constructible_v<_Tp> && is_move_assignable_v<_Tp>
void std::swap (
    _Tp & __a,
    _Tp & __b) [inline], [constexpr], [noexcept]
```

Swaps two values.

**Parameters**

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

**Returns**

Nothing.

**swap() [4/5]**

```
template<typename _Tp, size_t _Nm>
requires is_swappable_v<_Tp>
void std::swap (
    _Tp(&) __a[_Nm],
    _Tp(&) __b[_Nm]) [inline], [constexpr], [noexcept]
```

Swap the contents of two arrays.

**swap() [5/5]**

```
template<typename... _Elements>
enable_if< __and< __is_swappable< _Elements >... >::value >::type std::swap (
    tuple< _Elements... > & __x,
    tuple< _Elements... > & __y) [inline], [constexpr], [delete], [noexcept]
```

Exchange the values of two tuples.

**tie()**

```
template<typename... _Elements>
tuple<_Elements &... > std::tie (
    _Elements &... __args) [constexpr], [noexcept]
```

Return a tuple of lvalue references bound to the arguments.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits >::copyfmt\(\)](#).

**tuple\_cat()**

```
template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>&
::value>::type>
auto std::tuple_cat (
    _Tpls &&... __tpls) -> typename __tuple_cat_result<_Tpls...>::__type [constexpr]
```

Create a tuple containing all elements from multiple tuple-like objects.

**3.15.3 Variable Documentation****piecewise\_construct**

```
piecewise_construct_t std::piecewise_construct [inline], [constexpr]
```

Tag for piecewise construction of std::pair objects.

Referenced by [std::map<\\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, \\_Tp > > >::insert\\_or\\_assign\(\)](#),

[std::map<\\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, \\_Tp > > >::insert\\_or\\_assign\(\)](#), and [std::map<\\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, \\_Tp > > >::insert\\_or\\_assign\(\)](#).

### 3.15.4 Function Objects

Collaboration diagram for Function Objects:



#### Topics

- [Adaptors for pointers to functions](#)
- [Adaptors for pointers to members](#)
- [Arithmetic Function Object Classes](#)
- [Binder Classes](#)
- [Boolean Operations Classes](#)
- [Comparison Classes](#)
- [Hashes](#)
- [Negators](#)

#### Classes

- `struct std::binary_function< _Arg1, _Arg2, _Result >`
- `class std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- `class std::function< _Res(_ArgTypes...)>`
- `class std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`

- class `std::move_only_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- class `std::reference_wrapper<_Tp>`
- struct `std::unary_function<_Arg, _Result>`

## Functions

- template<typename \_Tp, typename \_Class>  
constexpr `_Mem_fn<_Tp _Class::*>` `std::mem_fn` (`_Tp _Class::*_pm`) noexcept

### 3.15.4.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables. Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform (a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will usually be inlined directly.

An *adaptable function object* is one which provides nested typedefs `result_type` and either `argument_type` (for a unary function) or `first_argument_type` and `second_argument_type` (for a binary function). Those typedefs are used by function object adaptors such as `bind2nd`. The standard library provides two class templates, `unary_function` and `binary_function`, which define those typedefs and so can be used as base classes of adaptable function objects.

Since C++11 the use of function object adaptors has been superseded by more powerful tools such as lambda expressions, `function<>`, and more powerful type deduction (using `auto` and `decltype`). The helpers for defining adaptable function objects are deprecated since C++11, and no longer part of the standard library since C++17. However, they are still defined and used by libstdc++ after C++17, as a conforming extension.

### 3.15.4.2 Function Documentation

#### `mem_fn()`

```
template<typename _Tp, typename _Class>
_Mem_fn<_Tp _Class::*> std::mem_fn (
    _Tp _Class::*_pm) [inline], [constexpr], [noexcept]
```

Returns a function object that forwards to the member pointer pointer `pm`.

This allows a pointer-to-member to be transformed into a function object that can be called with an object expression as its first argument.

For a pointer-to-data-member the result must be called with exactly one argument, the object expression that would be used as the first operand in a `obj.*memptr` or `objp->*memptr` expression.

For a pointer-to-member-function the result must be called with an object expression and any additional arguments to pass to the member function, as in an expression like `(obj.*memfun)(args...)` or `(objp->*memfun)(args...)`.

The object expression can be a pointer, reference, `reference_wrapper`, or smart pointer, and the call wrapper will dereference it as needed to apply the pointer-to-member.

Since

C++11

### 3.15.4.3 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



#### Classes

- class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`
- class `std::pointer_to_unary_function< _Arg, _Result >`

#### Functions

- template<typename \_Arg, typename \_Result>  
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun ( _Result(*__x)(_Arg))`
- template<typename \_Arg1, typename \_Arg2, typename \_Result>  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun ( _Result(*__x)(_Arg1, _Arg2))`

#### 3.15.4.3.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes `unary_function` and `binary_function`). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor `pointer_to_unary_function` is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor `pointer_to_binary_function` does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

**Deprecated** Deprecated in C++11, no longer in the standard since C++17.

#### 3.15.4.3.2 Function Documentation

`ptr_fun()` [1/2]

```

template<typename _Arg, typename _Result>
pointer_to_unary_function< _Arg, _Result > std::ptr_fun (
    _Result(* __x )(_Arg)) [inline]
  
```

One of the [adaptors for function pointers](#).

References [ptr\\_fun\(\)](#).

Referenced by [ptr\\_fun\(\)](#), and [ptr\\_fun\(\)](#).



**ptr\_fun()** [2/2]

```
template<typename _Arg1, typename _Arg2, typename _Result>
pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (
    _Result (* __x ) (_Arg1, _Arg2)) [inline]
```

One of the [adaptors for function pointers](#).

References [ptr\\_fun\(\)](#).

**3.15.4.4 Adaptors for pointers to members**

Collaboration diagram for Adaptors for pointers to members:

**Classes**

- class [std::const\\_mem\\_fun1\\_ref\\_t< \\_Ret, \\_Tp, \\_Arg >](#)
- class [std::const\\_mem\\_fun1\\_t< \\_Ret, \\_Tp, \\_Arg >](#)
- class [std::const\\_mem\\_fun\\_ref\\_t< \\_Ret, \\_Tp >](#)
- class [std::const\\_mem\\_fun\\_t< \\_Ret, \\_Tp >](#)
- class [std::mem\\_fun1\\_ref\\_t< \\_Ret, \\_Tp, \\_Arg >](#)
- class [std::mem\\_fun1\\_t< \\_Ret, \\_Tp, \\_Arg >](#)
- class [std::mem\\_fun\\_ref\\_t< \\_Ret, \\_Tp >](#)
- class [std::mem\\_fun\\_t< \\_Ret, \\_Tp >](#)

**Functions**

- template<typename \_Ret, typename \_Tp>  
[mem\\_fun\\_t](#)< \_Ret, \_Tp > **std::mem\_fun** (\_Ret(\_Tp::\*\_\_f)())
- template<typename \_Ret, typename \_Tp, typename \_Arg>  
[mem\\_fun1\\_t](#)< \_Ret, \_Tp, \_Arg > **std::mem\_fun** (\_Ret(\_Tp::\*\_\_f)(\_Arg))
- template<typename \_Ret, typename \_Tp>  
[mem\\_fun\\_ref\\_t](#)< \_Ret, \_Tp > **std::mem\_fun\_ref** (\_Ret(\_Tp::\*\_\_f)())
- template<typename \_Ret, typename \_Tp, typename \_Arg>  
[mem\\_fun1\\_ref\\_t](#)< \_Ret, \_Tp, \_Arg > **std::mem\_fun\_ref** (\_Ret(\_Tp::\*\_\_f)(\_Arg))

**3.15.4.4.1 Detailed Description**

There are a total of  $8 = 2^3$  function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

**Deprecated** Deprecated in C++11, no longer in the standard since C++17. Use `mem_fn` instead.

### 3.15.4.5 Arithmetic Function Object Classes

Collaboration diagram for Arithmetic Function Object Classes:



#### Classes

- struct [std::divides<\\_Tp>](#)
- struct [std::divides<void>](#)
- struct [std::minus<\\_Tp>](#)
- struct [std::minus<void>](#)
- struct [std::modulus<\\_Tp>](#)
- struct [std::modulus<void>](#)
- struct [std::multiplies<\\_Tp>](#)
- struct [std::multiplies<void>](#)
- struct [std::negate<\\_Tp>](#)
- struct [std::negate<void>](#)
- struct [std::plus<\\_Tp>](#)

#### 3.15.4.5.1 Detailed Description

The library provides function objects for basic arithmetic operations. See the documentation for [function objects](#) for examples of their use.

### 3.15.4.6 Binder Classes

Collaboration diagram for Binder Classes:



#### Namespaces

- namespace [std::placeholders](#)

## Classes

- struct `std::_Placeholder<_Num>`
- class `std::binder1st<_Operation>`
- class `std::binder2nd<_Operation>`
- struct `std::is_bind_expression<_Tp>`
- struct `std::is_bind_expression<_Bind<_Signature>>`
- struct `std::is_bind_expression<_Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<const _Bind<_Signature>>`
- struct `std::is_bind_expression<const _Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<const volatile _Bind<_Signature>>`
- struct `std::is_bind_expression<const volatile _Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<volatile _Bind<_Signature>>`
- struct `std::is_bind_expression<volatile _Bind_result<_Result, _Signature>>`
- struct `std::is_placeholder<_Tp>`
- struct `std::is_placeholder<_Placeholder<_Num>>`

## Functions

- template<typename \_Func, typename... \_BoundArgs>  
constexpr `_Bind_helper<__is_socketlike<_Func>::value, _Func, _BoundArgs...>::type` `std::bind` (`_Func` &&\_\_f, `_BoundArgs` &&... \_\_args)
- template<typename \_Result, typename \_Func, typename... \_BoundArgs>  
constexpr `_Bindres_helper<_Result, _Func, _BoundArgs...>::type` `std::bind` (`_Func` &&\_\_f, `_BoundArgs` &&... \_\_args)
- template<typename \_Operation, typename \_Tp>  
`binder1st<_Operation>` `std::bind1st` (`const _Operation` &\_\_fn, `const _Tp` &\_\_x)
- template<typename \_Operation, typename \_Tp>  
`binder2nd<_Operation>` `std::bind2nd` (`const _Operation` &\_\_fn, `const _Tp` &\_\_x)

### 3.15.4.6.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, `B`'s `operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be *called* with various arguments (`y1`, `y2`, ...) and will in turn call `f(x, y1)`, `f(x, y2)`, ...

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>(), 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform `1.3 - x` instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st(std::plus<int>(), 5)`).

These become more useful when combined with the composition functions.

These functions are deprecated in C++11 and can be replaced by `std::bind` (or `std::tr1::bind`) which is more powerful and flexible, supporting functions with any number of arguments. Uses of `bind1st` can be replaced by `std::bind(f, x, std::placeholders::_1)` and `bind2nd` by `std::bind(f, std::placeholders::_1, x)`.

### 3.15.4.6.2 Function Documentation

#### bind() [1/2]

```
template<typename _Func, typename... _BoundArgs>
_Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type std::bind (
    _Func && __f,
    _BoundArgs &&... __args) [inline], [constexpr]
```

Function template for `std::bind`.

Since

C++11

#### bind() [2/2]

```
template<typename _Result, typename _Func, typename... _BoundArgs>
_Bindres_helper< _Result, _Func, _BoundArgs... >::type std::bind (
    _Func && __f,
    _BoundArgs &&... __args) [inline], [constexpr]
```

Function template for `std::bind<R>`.

Since

C++11

#### bind1st()

```
template<typename _Operation, typename _Tp>
binder1st< _Operation > std::bind1st (
    const _Operation & __fn,
    const _Tp & __x) [inline]
```

One of the [binder functors](#).

References [bind1st\(\)](#).

Referenced by [bind1st\(\)](#).

#### bind2nd()

```
template<typename _Operation, typename _Tp>
binder2nd< _Operation > std::bind2nd (
    const _Operation & __fn,
    const _Tp & __x) [inline]
```

One of the [binder functors](#).

References [bind2nd\(\)](#).

Referenced by [bind2nd\(\)](#).

### 3.15.4.7 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



#### Classes

- struct `std::logical_and< _Tp >`
- struct `std::logical_and< void >`
- struct `std::logical_not< _Tp >`
- struct `std::logical_not< void >`
- struct `std::logical_or< _Tp >`
- struct `std::logical_or< void >`

#### 3.15.4.7.1 Detailed Description

The library provides function objects for the logical operations: `&&`, `||`, and `!`.

### 3.15.4.8 Comparison Classes

Collaboration diagram for Comparison Classes:



#### Classes

- struct `std::equal_to< _Tp >`
- struct `std::equal_to< void >`
- struct `std::greater< _Tp >`
- struct `std::greater< void >`
- struct `std::greater_equal< _Tp >`
- struct `std::greater_equal< void >`
- struct `std::less< _Tp >`
- struct `std::less< void >`
- struct `std::less_equal< _Tp >`

- struct `std::less_equal< void >`
- struct `std::not_equal_to< _Tp >`
- struct `std::not_equal_to< void >`

#### 3.15.4.8.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

#### 3.15.4.9 Hashes

Collaboration diagram for Hashes:



#### Classes

- struct `std::hash< _Tp >`
- struct `std::hash< _Tp * >`
- struct `std::hash< bool >`
- struct `std::hash< char >`
- struct `std::hash< char16_t >`
- struct `std::hash< char32_t >`
- struct `std::hash< double >`
- struct `std::hash< float >`
- struct `std::hash< int >`
- struct `std::hash< long >`
- struct `std::hash< long double >`
- struct `std::hash< long long >`
- struct `std::hash< short >`
- struct `std::hash< signed char >`
- struct `std::hash< unsigned char >`
- struct `std::hash< unsigned int >`
- struct `std::hash< unsigned long >`
- struct `std::hash< unsigned long long >`
- struct `std::hash< unsigned short >`
- struct `std::hash< wchar_t >`

#### Macros

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

## Variables

- `template<typename _Tp, typename = void>`  
`constexpr bool std::__is_hash_enabled_for`
- `template<typename _Tp>`  
`constexpr bool std::__is_hash_enabled_for< _Tp, __void_t< decltype(hash< _Tp>{}(declval< _Tp>{}))>`  
`>`

### 3.15.4.9.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

### 3.15.4.10 Negators

Collaboration diagram for Negators:



## Classes

- class `std::binary_negate< _Predicate >`
- class `std::unary_negate< _Predicate >`

## Functions

- `template<typename _Predicate>`  
`constexpr unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate>`  
`constexpr binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`

### 3.15.4.10.1 Detailed Description

The function templates `not1` and `not2` are function object adaptors, which each take a predicate functor and wrap it in an instance of `unary_negate` or `binary_negate`, respectively. Those classes are functors whose `operator()` evaluates the wrapped predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
    bool operator() (int x) const { return x > 3; }
};

std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
  
```

The call to `find_if` will locate the first index (*i*) of *v* for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates taking two arguments.

**Deprecated** Deprecated in C++17, no longer in the standard since C++20. Use `not_fn` instead.

### 3.15.4.10.2 Function Documentation

#### not1()

```
template<typename _Predicate>
unary_negate< _Predicate > std::not1 (
    const _Predicate & __pred) [inline], [constexpr]
```

One of the [negation functors](#).

References [not1\(\)](#).

Referenced by [not1\(\)](#).

#### not2()

```
template<typename _Predicate>
binary_negate< _Predicate > std::not2 (
    const _Predicate & __pred) [inline], [constexpr]
```

One of the [negation functors](#).

References [not2\(\)](#).

Referenced by [not2\(\)](#).

## 3.15.5 Memory

Collaboration diagram for Memory:



### Topics

- [Allocators](#)
- [Pointer Abstractions](#)
- [Pointer Safety and Garbage Collection](#)
- [Polymorphic memory resources](#)

### Files

- file [memory](#)



## Functions

- `void * std::align (size_t __align, size_t __size, void *& __ptr, size_t & __space) noexcept`
- `template<size_t Align, class _Tp>  
constexpr _Tp * std::assume\_aligned (_Tp * __ptr) noexcept`
- `template<typename _InputIterator, typename _ForwardIterator>  
_ForwardIterator std::uninitialized\_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>  
_ForwardIterator std::uninitialized\_copy\_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator>  
void std::uninitialized\_default\_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>  
_ForwardIterator std::uninitialized\_default\_construct\_n (_ForwardIterator __first, _Size __count)`
- `template<typename _ForwardIterator, typename _Tp>  
void std::uninitialized\_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp>  
_ForwardIterator std::uninitialized\_fill\_n (_ForwardIterator __first, _Size __n, const _Tp & __x)`
- `template<typename _InputIterator, typename _ForwardIterator>  
_ForwardIterator std::uninitialized\_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>  
pair< _InputIterator, _ForwardIterator > std::uninitialized\_move\_n (_InputIterator __first, _Size __count, _ForwardIterator __result)`
- `template<typename _ForwardIterator>  
void std::uninitialized\_value\_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>  
_ForwardIterator std::uninitialized\_value\_construct\_n (_ForwardIterator __first, _Size __count)`

### 3.15.5.1 Detailed Description

Components for memory allocation, deallocation, and management.

### 3.15.5.2 Function Documentation

#### `align()`

```
void * std::align (
    size_t __align,
    size_t __size,
    void *& __ptr,
    size_t & __space) [inline], [noexcept]
```

Fit aligned storage in buffer.

This function tries to fit `__size` bytes of storage with alignment `__align` into the buffer `__ptr` of size `__space` bytes. If such a buffer fits then `__ptr` is changed to point to the first byte of the aligned storage and `__space` is reduced by the bytes used for alignment.

C++11 20.6.5 [ptr.align]

#### Parameters

<code>__align</code>	A fundamental or extended alignment value.
<code>__size</code>	Size of the aligned storage required.
<code>__ptr</code>	Pointer to a buffer of <code>__space</code> bytes.
<code>__space</code>	Size of the buffer pointed to by <code>__ptr</code> .

**Returns**

the updated pointer if the aligned storage fits, otherwise nullptr.

**assume\_aligned()**

```
template<size_t _Align, class _Tp>
_Tp * std::assume_aligned (
    _Tp * __ptr) [nodiscard], [constexpr], [noexcept]
```

Inform the compiler that a pointer is aligned.

**Template Parameters**

<code>_Align</code>	An alignment value (i.e. a power of two)
<code>_Tp</code>	An object type

**Parameters**

<code>__ptr</code>	A pointer that is aligned to <code>_Align</code>
--------------------	--

C++20 20.10.6 [ptr.align]

**uninitialized\_copy()**

```
template<typename _InputIterator, typename _ForwardIterator>
_FowardIterator std::uninitialized_copy (
    _InputIterator __first,
    _InputIterator __last,
    _ForwardIterator __result) [inline]
```

Copies the range [first,last) into result.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	A forward iterator.

**Returns**

`__result + (__last - __first)`

Like `std::copy`, but does not require an initialized output range.

References [to\\_address\(\)](#).

Referenced by [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#), and [uninitialized\\_move\(\)](#).

**uninitialized\_copy\_n()**

```
template<typename _InputIterator, typename _Size, typename _ForwardIterator>
_FowardIterator std::uninitialized_copy_n (
    _InputIterator __first,
    _Size __n,
    _ForwardIterator __result) [inline]
```

Copies the range [first,first+n) into result.

**Parameters**

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

**Returns**

`__result + __n`

**Since**

C++11

Like `copy_n()`, but does not require an initialized output range.

References [\\_\\_iterator\\_category\(\)](#).

**uninitialized\_default\_construct()**

```
template<typename _ForwardIterator>
void std::uninitialized_default_construct (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline]
```

Default-initializes objects in the range `[first,last)`.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Since**

C++17

**uninitialized\_default\_construct\_n()**

```
template<typename _ForwardIterator, typename _Size>
_FowardIterator std::uninitialized_default_construct_n (
    _ForwardIterator __first,
    _Size __count) [inline]
```

Default-initializes objects in the range `[first,first+count)`.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__count</code>	The number of objects to construct.

**Returns**

`__first + __count`

**Since**

C++17

**uninitialized\_fill()**

```
template<typename _ForwardIterator, typename _Tp>
void std::uninitialized_fill (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __x) [inline]
```

Copies the value x into the range [first,last).

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__x</code>	The source value.

**Returns**

Nothing.

Like `std::fill`, but does not require an initialized output range.

References [to\\_address\(\)](#).

**uninitialized\_fill\_n()**

```
template<typename _ForwardIterator, typename _Size, typename _Tp>
_FForwardIterator std::uninitialized_fill_n (
    _ForwardIterator __first,
    _Size __n,
    const _Tp & __x) [inline]
```

Copies the value x into the range [first,first+n).

**Parameters**

<code>__first</code>	A forward iterator.
<code>__n</code>	The number of copies to make.
<code>__x</code>	The source value.

**Returns**

`__first + __n`.

Like `std::fill_n`, but does not require an initialized output range.

References [to\\_address\(\)](#).

**uninitialized\_move()**

```
template<typename _InputIterator, typename _ForwardIterator>
_FForwardIterator std::uninitialized_move (
    _InputIterator __first,
    _InputIterator __last,
    _ForwardIterator __result) [inline]
```

Move-construct from the range [first,last) into result.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

**Returns**

`__result + (__first - __last)`

**Since**

C++17

References [uninitialized\\_copy\(\)](#).

**uninitialized\_move\_n()**

```
template<typename _InputIterator, typename _Size, typename _ForwardIterator>
pair< _InputIterator, _ForwardIterator > std::uninitialized_move_n (
    _InputIterator __first,
    _Size __count,
    _ForwardIterator __result) [inline]
```

Move-construct from the range [first,first+count) into result.

**Parameters**

<code>__first</code>	An input iterator.
<code>__count</code>	The number of objects to initialize.
<code>__result</code>	An output iterator.

**Returns**

`__result + __count`

**Since**

C++17

**uninitialized\_value\_construct()**

```
template<typename _ForwardIterator>
void std::uninitialized_value_construct (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline]
```

Value-initializes objects in the range [first,last).

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Since**

C++17

**uninitialized\_value\_construct\_n()**

```
template<typename _ForwardIterator, typename _Size>
_FowardIterator std::uninitialized_value_construct_n (
    _ForwardIterator __first,
    _Size __count) [inline]
```

Value-initializes objects in the range [first,first+count).

**Parameters**

<code>__first</code>	A forward iterator.
<code>__count</code>	The number of objects to construct.

**Returns**

`__result + __count`

**Since**

C++17

**3.15.5.3 Allocators**

Collaboration diagram for Allocators:

**Classes**

- struct `__gnu_cxx::__alloc_traits< _Alloc, typename >`
- class `__gnu_cxx::__mt_alloc< _Tp, _Poolp >`
- class `std::__new_allocator< _Tp >`
- class `__gnu_cxx::__pool_alloc< _Tp >`
- class `__gnu_cxx::__ExtPtr_allocator< _Tp >`
- class `std::allocator< _Tp >`
- struct `std::allocator_traits< _Alloc >`
- struct `std::allocator_traits< allocator< _Tp > >`
- struct `std::allocator_traits< allocator< void > >`
- class `__gnu_cxx::bitmap_allocator< _Tp >`
- class `__gnu_cxx::debug_allocator< _Alloc >`
- class `__gnu_cxx::malloc_allocator< _Tp >`
- class `__gnu_cxx::new_allocator< _Tp >`
- class `std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >`
- class `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`
- struct `std::uses_allocator< typename, typename >`

## Typedefs

- `template<typename _Tp>`  
using `std::__allocator_base`

## Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`  
constexpr `_Tp` **std::make\_obj\_using\_allocator** (const `_Alloc` &\_\_a, `_Args` &&... \_\_args)
- `template<typename _T1, typename _T2>`  
constexpr bool **operator==** (const `allocator`< `_T1` > &, const `allocator`< `_T2` > &) noexcept
- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
bool **operator==** (const `scoped_allocator_adaptor`< `_OutA1`, `_InA`... > &\_\_a, const `scoped_allocator_adaptor`< `_OutA2`, `_InA`... > &\_\_b) noexcept
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
constexpr `_Tp` \* **std::uninitialized\_construct\_using\_allocator** (`_Tp` \*\_\_p, const `_Alloc` &\_\_a, `_Args` &&... \_\_args)
- `template<_Std_pair _Tp, typename _Alloc>`  
constexpr auto **std::uses\_allocator\_construction\_args** (const `_Alloc` &) noexcept
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
constexpr auto **std::uses\_allocator\_construction\_args** (const `_Alloc` &, `_Up` &&, `_Vp` &&) noexcept
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
constexpr auto **std::uses\_allocator\_construction\_args** (const `_Alloc` &, const `pair`< `_Up`, `_Vp` > &) noexcept
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
constexpr auto **std::uses\_allocator\_construction\_args** (const `_Alloc` &, `pair`< `_Up`, `_Vp` > &&) noexcept
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
requires (! `_Std_pair`< `_Tp`>)  
constexpr auto **std::uses\_allocator\_construction\_args** (const `_Alloc` &\_\_a, `_Args` &&... \_\_args) noexcept
- `template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>`  
constexpr auto **std::uses\_allocator\_construction\_args** (const `_Alloc` &\_\_a, `piecewise_construct_t`, `_Tuple1` &&\_\_x, `_Tuple2` &&\_\_y) noexcept

### 3.15.5.3.1 Detailed Description

Classes encapsulating memory operations.

### 3.15.5.3.2 Typedef Documentation

#### `__allocator_base`

```
template<typename _Tp>
using std::__allocator_base
```

An alias to the base class for `std::allocator`.

Used to set the `std::allocator` base class to `std::__new_allocator`.

#### Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

### 3.15.5.3.3 Function Documentation

#### **operator==( )**

```
template<typename _T1, typename _T2>
bool operator== (
```

```
const allocator< _T1 > & ,
const allocator< _T2 > & ) [related]
```

Equality comparison for `std::allocator` objects

Returns

true, for all `std::allocator` objects.

### 3.15.5.4 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



## Classes

- class `std::auto_ptr< _Tp >`
- struct `std::default_delete< _Tp >`
- struct `std::default_delete< _Tp[] >`
- class `std::enable_shared_from_this< _Tp >`
- struct `std::hash< shared_ptr< _Tp > >`
- struct `std::hash< unique_ptr< _Tp, _Dp > >`
- struct `std::owner_less< _Tp >`
- struct `std::owner_less< shared_ptr< _Tp > >`
- struct `std::owner_less< void >`
- struct `std::owner_less< weak_ptr< _Tp > >`
- struct `std::pointer_traits< _Ptr >`
- struct `std::pointer_traits< _Tp * >`
- class `std::shared_ptr< _Tp >`
- class `std::unique_ptr< _Tp, _Dp >`
- class `std::unique_ptr< _Tp[], _Dp >`
- class `std::weak_ptr< _Tp >`

## Functions

- template<typename \_Del, typename \_Tp, \_Lock\_policy \_Lp>  
\_Del \* **std::get\_deleter** (const \_\_shared\_ptr< \_Tp, \_Lp > &\_\_p) noexcept
- template<typename \_Del, typename \_Tp>  
\_Del \* **get\_deleter** (const `shared_ptr`< \_Tp > &\_\_p) noexcept
- template<typename \_Tp, typename \_Dp, typename \_Up, typename \_Ep>  
constexpr bool **std::operator<** (const `unique_ptr`< \_Tp, \_Dp > &\_\_x, const `unique_ptr`< \_Up, \_Ep > &\_\_y)
- template<typename \_Tp, typename \_Dp>  
constexpr bool **std::operator<** (const `unique_ptr`< \_Tp, \_Dp > &\_\_x, nullptr\_t)
- template<typename \_Tp, typename \_Dp>  
constexpr bool **std::operator<** (nullptr\_t, const `unique_ptr`< \_Tp, \_Dp > &\_\_x)



- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`requires three_way_comparable_with<typename unique_ptr<_Tp, _Dp>::pointer, typename unique_ptr<_Up, _Ep>::pointer>`  
`constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > std::operator<=> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`requires three_way_comparable<typename unique_ptr<_Tp, _Dp>::pointer>`  
`constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > std::operator<=> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Del>`  
`std::shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >`
- `template<typename _Tp>`  
`std::shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >`
- `template<typename _Tp, typename _Dp>`  
`enable_if<! __is_swappable< _Dp >::value >::type std::swap (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete`
- `template<typename _Tp, typename _Dp>`  
`constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Tp>`  
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp * std::to_address (_Tp *__ptr) noexcept`
- `template<typename _Ptr>`  
`constexpr auto std::to_address (const _Ptr &__ptr) noexcept`

- `template<typename _Tp>`  
`std::weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)`
- `template<typename _Tp>`  
`bool std::atomic_is_lock_free (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_load (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_load (const __shared_ptr< _Tp, _Lp > *__p)`
- `template<typename _Tp>`  
`void std::atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
- `template<typename _Tp>`  
`void std::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)`

- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *↔`  
`__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp`  
`> *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v,`  
`__shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp, typename _Up>`  
`bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp>`  
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > static_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > const_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > dynamic_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr< _NonArray< _Tp > > allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, typename... _Args>`  
`shared_ptr< _NonArray< _Tp > > make_shared (_Args &&... __args)`

### 3.15.5.4.1 Detailed Description

Smart pointers, etc.

### 3.15.5.4.2 Function Documentation

#### `allocate_shared()`

```
template<typename _Tp, typename _Alloc, typename... _Args>
shared_ptr< _NonArray< _Tp > > allocate_shared (
    const _Alloc & __a,
    _Args &&... __args) [related]
```

Create an object that is owned by a `shared_ptr`.

#### Parameters

<code>__a</code>	An allocator.
------------------	---------------

<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.
---------------------	--

#### Returns

A `shared_ptr` that owns the newly created object.

#### Exceptions

<i>An</i>	exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> .
-----------	---

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

#### `atomic_compare_exchange_strong()` [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > * __v,
    __shared_ptr< _Tp, _Lp > __w) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

#### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

#### Returns

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.  
References [move\(\)](#).

#### `atomic_compare_exchange_strong()` [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_strong (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > * __v,
    shared_ptr< _Tp > __w) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

#### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.

<a href="#"></a>	A non-null pointer to a shared_ptr object.
<a href="#"></a> _w	

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.

References [move\(\)](#).

**atomic\_compare\_exchange\_strong\_explicit() [1/2]**

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong_explicit (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > * __v,
    __shared_ptr< _Tp, _Lp > __w,
    memory_order ,
    memory_order )
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

<a href="#"></a> _p	A non-null pointer to a shared_ptr object.
<a href="#"></a> _v	A non-null pointer to a shared_ptr object.
<a href="#"></a> _w	A non-null pointer to a shared_ptr object.

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.

References [move\(\)](#).

**atomic\_compare\_exchange\_strong\_explicit() [2/2]**

```
template<typename _Tp>
bool std::atomic_compare_exchange_strong_explicit (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > * __v,
    shared_ptr< _Tp > __w,
    memory_order ,
    memory_order )
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

<a href="#"></a> _p	A non-null pointer to a shared_ptr object.
<a href="#"></a> _v	A non-null pointer to a shared_ptr object.

<a href="#"></a>	A non-null pointer to a shared_ptr object.
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.  
References [move\(\)](#).

**atomic\_compare\_exchange\_weak() [1/2]**

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > * __v,
    __shared_ptr< _Tp, _Lp > __w) [inline]
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

<a href="#"></a>	A non-null pointer to a shared_ptr object.
<a href="#"></a>	
<a href="#"></a>	A non-null pointer to a shared_ptr object.
<a href="#"></a>	
<a href="#"></a>	A non-null pointer to a shared_ptr object.
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.  
References [move\(\)](#).

**atomic\_compare\_exchange\_weak() [2/2]**

```
template<typename _Tp>
bool std::atomic_compare_exchange_weak (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > * __v,
    shared_ptr< _Tp > __w) [inline]
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

<a href="#"></a>	A non-null pointer to a shared_ptr object.
<a href="#"></a>	
<a href="#"></a>	A non-null pointer to a shared_ptr object.
<a href="#"></a>	
<a href="#"></a>	A non-null pointer to a shared_ptr object.
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	
<a href="#"></a>	

**Returns**

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.  
References [move\(\)](#).

**atomic\_compare\_exchange\_weak\_explicit() [1/2]**

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak_explicit (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > * __v,
    __shared_ptr< _Tp, _Lp > __w,
    memory_order __success,
    memory_order __failure) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

**Parameters**

 <code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
 <code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
 <code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

**Returns**

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.  
References [move\(\)](#).

**atomic\_compare\_exchange\_weak\_explicit() [2/2]**

```
template<typename _Tp>
bool std::atomic_compare_exchange_weak_explicit (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > * __v,
    shared_ptr< _Tp > __w,
    memory_order __success,
    memory_order __failure) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

**Parameters**

 <code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
 <code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
 <code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

**Returns**

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.

References [move\(\)](#).

**atomic\_exchange()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_exchange (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > __r) [inline]
```

Atomic exchange for `shared_ptr` objects.

**Parameters**

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__r</code>	New value to store in <code>*__p</code> .

**Returns**

The original value of `*__p`

References [move\(\)](#).

**atomic\_exchange()** [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_exchange (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > __r) [inline]
```

Atomic exchange for `shared_ptr` objects.

**Parameters**

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__r</code>	New value to store in <code>*__p</code> .

**Returns**

The original value of `*__p`

References [move\(\)](#).

**atomic\_exchange\_explicit()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > __r,
    memory_order ) [inline]
```

Atomic exchange for `shared_ptr` objects.



**Parameters**

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
$\leftarrow$ _r	New value to store in *__p.

**Returns**

The original value of \*\_\_p

**atomic\_exchange\_explicit()** [2/2]

```
template<typename _Tp>
shared_ptr<_Tp> std::atomic_exchange_explicit (
    shared_ptr<_Tp> * __p,
    shared_ptr<_Tp> __r,
    memory_order ) [inline]
```

Atomic exchange for shared\_ptr objects.

**Parameters**

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
$\leftarrow$ _r	New value to store in *__p.

**Returns**

The original value of \*\_\_p

**atomic\_is\_lock\_free()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_is_lock_free (
    const __shared_ptr<_Tp, _Lp> * ) [inline]
```

Report whether shared\_ptr atomic operations are lock-free.

**Parameters**

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
--------------------	--

**Returns**

True if atomic access to \*\_\_p is lock-free, false otherwise.

**atomic\_is\_lock\_free()** [2/2]

```
template<typename _Tp>
bool std::atomic_is_lock_free (
    const shared_ptr<_Tp> * __p) [inline]
```

Report whether shared\_ptr atomic operations are lock-free.

## Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
------------------	---

## Returns

True if atomic access to `*__p` is lock-free, false otherwise.

**atomic\_load()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_load (
    const __shared_ptr< _Tp, _Lp > * __p) [inline]
```

Atomic load for `shared_ptr` objects.

## Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
------------------	---

## Returns

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

**atomic\_load()** [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_load (
    const shared_ptr< _Tp > * __p) [inline]
```

Atomic load for `shared_ptr` objects.

## Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
------------------	---

## Returns

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

**atomic\_load\_explicit()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_load_explicit (
    const __shared_ptr< _Tp, _Lp > * __p,
    memory_order ) [inline]
```

Atomic load for `shared_ptr` objects.

**Parameters**

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
--------------------	--

**Returns**

\*\_\_p

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

**atomic\_load\_explicit() [2/2]**

```
template<typename _Tp>
shared_ptr<_Tp> std::atomic_load_explicit (
    const shared_ptr<_Tp> * __p,
    memory_order ) [inline]
```

Atomic load for shared\_ptr objects.

**Parameters**

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
--------------------	--

**Returns**

\*\_\_p

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

**atomic\_store() [1/2]**

```
template<typename _Tp, _Lock_policy _Lp>
void std::atomic_store (
    __shared_ptr<_Tp, _Lp> * __p,
    __shared_ptr<_Tp, _Lp> __r) [inline]
```

Atomic store for shared\_ptr objects.

**Parameters**

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
$\leftarrow$ _r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

References [move\(\)](#).

**atomic\_store() [2/2]**

```
template<typename _Tp>
void std::atomic_store (
    shared_ptr<_Tp> * __p,
    shared_ptr<_Tp> __r) [inline]
```

Atomic store for shared\_ptr objects.

## Parameters

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
$\leftarrow$ _r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.  
References [move\(\)](#).

**atomic\_store\_explicit()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
void std::atomic_store_explicit (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > __r,
    memory_order ) [inline]
```

Atomic store for shared\_ptr objects.

## Parameters

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
$\leftarrow$ _r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

**atomic\_store\_explicit()** [2/2]

```
template<typename _Tp>
void std::atomic_store_explicit (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > __r,
    memory_order ) [inline]
```

Atomic store for shared\_ptr objects.

## Parameters

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
$\leftarrow$ _r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

**const\_pointer\_cast()** [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > const_pointer_cast (
    const shared_ptr< _Up > & __r) [related]
```

Convert type of shared\_ptr, via const\_cast

**const\_pointer\_cast()** [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > const_pointer_cast (
    shared_ptr< _Up > && __r) [related]
```

Convert type of shared\_ptr rvalue, via const\_cast

Since

C++20

**dynamic\_pointer\_cast()** [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > dynamic_pointer_cast (
    const shared_ptr< _Up > & __r) [related]
```

Convert type of shared\_ptr, via dynamic\_cast

**dynamic\_pointer\_cast()** [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > dynamic_pointer_cast (
    shared_ptr< _Up > && __r) [related]
```

Convert type of shared\_ptr rvalue, via dynamic\_cast

Since

C++20

**get\_deleter()**

```
template<typename _Del, typename _Tp>
_Del * get_deleter (
    const shared_ptr< _Tp > & __p) [related]
```

20.7.2.2.10 shared\_ptr get\_deleter

If \_\_p has a deleter of type \_Del, return a pointer to it.

**make\_shared()**

```
template<typename _Tp, typename... _Args>
shared_ptr< _NonArray< _Tp > > make_shared (
    _Args &&... __args) [related]
```

Create an object that is owned by a shared\_ptr.

**Parameters**

<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.
---------------------	--

**Returns**

A shared\_ptr that owns the newly created object.

**Exceptions**

<code>std::bad_alloc</code> , or	an exception thrown from the constructor of <code>_Tp</code> .
----------------------------------	--

**operator<>() [1/3]**

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator< (
    const unique_ptr< _Tp, _Dp > & __x,
    const unique_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Relational operator for unique\_ptr objects, compares the owned pointers.

References [std::unique\\_ptr< \\_Tp, \\_Dp >::get\(\)](#).

**operator<>() [2/3]**

```
template<typename _Tp, typename _Dp>
bool std::operator< (
    const unique_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [inline], [nodiscard], [constexpr]
```

unique\_ptr comparison with nullptr

References [std::unique\\_ptr< \\_Tp, \\_Dp >::get\(\)](#).

**operator<>() [3/3]**

```
template<typename _Tp, typename _Dp>
bool std::operator< (
    nullptr_t ,
    const unique_ptr< _Tp, _Dp > & __x) [inline], [nodiscard], [constexpr]
```

unique\_ptr comparison with nullptr

References [std::unique\\_ptr< \\_Tp, \\_Dp >::get\(\)](#).

**operator<<() [1/2]**

```
template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
std::basic_ostream< _Ch, _Tr > & operator<< (
    std::basic_ostream< _Ch, _Tr > & __os,
    const __shared_ptr< _Tp, _Lp > & __p) [related]
```

Write the stored pointer to an ostream.

**operator<<() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
basic_ostream< _CharT, _Traits > & operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const unique_ptr< _Tp, _Dp > & __p) [related]
```

Stream output operator for unique\_ptr.

Since

C++20

**operator<=() [1/3]**

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator<= (
    const unique_ptr< _Tp, _Dp > & __x,
    const unique_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Relational operator for unique\_ptr objects, compares the owned pointers.

**operator<=()** [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator<= (
    const unique\_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [inline], [nodiscard], [constexpr]
```

[unique\\_ptr](#) comparison with [nullptr](#)

**operator<=()** [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator<= (
    nullptr_t ,
    const unique\_ptr< _Tp, _Dp > & __x) [inline], [nodiscard], [constexpr]
```

[unique\\_ptr](#) comparison with [nullptr](#)

**operator==()** [1/4]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator== (
    const unique\_ptr< _Tp, _Dp > & __x,
    const unique\_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Equality operator for [unique\\_ptr](#) objects, compares the owned pointers.

References [std::unique\\_ptr<\\_Tp, \\_Dp>::get\(\)](#).

**operator==()** [2/4]

```
template<typename _Tp, typename _Dp>
bool std::operator== (
    const unique\_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [inline], [nodiscard], [constexpr], [noexcept]
```

[unique\\_ptr](#) comparison with [nullptr](#)

**operator==()** [3/4]

```
template<typename _Tp, typename _Up>
bool operator== (
    const shared\_ptr< _Tp > & __a,
    const shared\_ptr< _Up > & __b) [related]
```

Equality operator for [shared\\_ptr](#) objects, compares the stored pointers.

**operator==()** [4/4]

```
template<typename _Tp>
bool operator== (
    const shared\_ptr< _Tp > & __a,
    nullptr_t ) [related]
```

[shared\\_ptr](#) comparison with [nullptr](#)

**operator>()** [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator> (
    const unique\_ptr< _Tp, _Dp > & __x,
    const unique\_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Relational operator for [unique\\_ptr](#) objects, compares the owned pointers.

**operator>()** [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator> (
    const unique\_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [inline], [nodiscard], [constexpr]
```

[unique\\_ptr](#) comparison with [nullptr](#)

References [std::unique\\_ptr< \\_Tp, \\_Dp >::get\(\)](#).

**operator>()** [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator> (
    nullptr_t ,
    const unique\_ptr< _Tp, _Dp > & __x) [inline], [nodiscard], [constexpr]
```

[unique\\_ptr](#) comparison with [nullptr](#)

References [std::unique\\_ptr< \\_Tp, \\_Dp >::get\(\)](#).

**operator>=()** [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator>= (
    const unique\_ptr< _Tp, _Dp > & __x,
    const unique\_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Relational operator for [unique\\_ptr](#) objects, compares the owned pointers.

**operator>=()** [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator>= (
    const unique\_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [inline], [nodiscard], [constexpr]
```

[unique\\_ptr](#) comparison with [nullptr](#)

**operator>=()** [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator>= (
    nullptr_t ,
    const unique\_ptr< _Tp, _Dp > & __x) [inline], [nodiscard]
```

[unique\\_ptr](#) comparison with [nullptr](#)

**reinterpret\_pointer\_cast()** [1/2]

```
template<typename _Tp, typename _Up>
shared\_ptr< _Tp > reinterpret_pointer_cast (
    const shared\_ptr< _Up > & __r) [related]
```

Convert type of [shared\\_ptr](#), via [reinterpret\\_cast](#)

Since

C++17



**reinterpret\_pointer\_cast()** [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > reinterpret_pointer_cast (
    shared_ptr< _Up > && __r) [related]
```

Convert type of shared\_ptr rvalue, via reinterpret\_cast

Since

C++20

**static\_pointer\_cast()** [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > static_pointer_cast (
    const shared_ptr< _Up > & __r) [related]
```

Convert type of shared\_ptr, via static\_cast

**static\_pointer\_cast()** [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > static_pointer_cast (
    shared_ptr< _Up > && __r) [related]
```

Convert type of shared\_ptr rvalue, via static\_cast

Since

C++20

**swap()** [1/3]

```
template<typename _Tp>
void swap (
    shared_ptr< _Tp > & __a,
    shared_ptr< _Tp > & __b) [related]
```

Swap overload for shared\_ptr.

**swap()** [2/3]

```
template<typename _Tp, typename _Dp>
enable_if< __is_swappable< _Dp >::value >::type swap (
    unique_ptr< _Tp, _Dp > & __x,
    unique_ptr< _Tp, _Dp > & __y) [related]
```

Swap overload for unique\_ptr.

**swap()** [3/3]

```
template<typename _Tp>
void swap (
    weak_ptr< _Tp > & __a,
    weak_ptr< _Tp > & __b) [related]
```

Swap overload for weak\_ptr.

**to\_address()** [1/2]

```
template<typename _Tp>
_Tp * std::to_address (
    _Tp * __ptr) [constexpr, [noexcept]]
```

Obtain address referenced by a pointer to an object.

**Parameters**

<code>__ptr</code>	A pointer to an object
--------------------	------------------------

**Returns**

`__ptr`

Referenced by [std::basic\\_string<\\_CharT>::assign\(\)](#), [find\(\)](#), [to\\_address\(\)](#), [uninitialized\\_copy\(\)](#), [uninitialized\\_fill\(\)](#), and [uninitialized\\_fill\\_n\(\)](#).

**to\_address()** [2/2]

```
template<typename _Ptr>
auto std::to_address (
    const _Ptr & __ptr) [constexpr, [noexcept]]
```

Obtain address referenced by a pointer to an object.

**Parameters**

<code>__ptr</code>	A pointer to an object
--------------------	------------------------

**Returns**

`pointer_traits<_Ptr>::to_address(__ptr)` if that expression is well-formed, otherwise `to_address(__ptr.operator->())`.

References [to\\_address\(\)](#).

**3.15.5.5 Pointer Safety and Garbage Collection**

Collaboration diagram for Pointer Safety and Garbage Collection:

**Enumerations**

- enum class [std::pointer\\_safety](#) { **relaxed** , **preferred** , **strict** }

## Functions

- void `std::declare_no_pointers` (char \*, size\_t)
- void `std::declare_reachable` (void \*)
- `pointer_safety` `std::get_pointer_safety` () noexcept
- void `std::undeclare_no_pointers` (char \*, size\_t)
- template<typename \_Tp>  
\_Tp \* `std::undeclare_reachable` (\_Tp \* \_\_p)

### 3.15.5.5.1 Detailed Description

Utilities to assist with garbage collection in an implementation that supports *strict pointer safety*. This implementation only supports *relaxed pointer safety* and so these functions have no effect.

C++11 20.6.4 [util.dynamic.safety], Pointer safety

### 3.15.5.5.2 Enumeration Type Documentation

#### `pointer_safety`

```
enum class std::pointer_safety [strong]
```

Constants representing the different types of pointer safety.

### 3.15.5.5.3 Function Documentation

#### `declare_no_pointers()`

```
void std::declare_no_pointers (  
    char * ,  
    size_t ) [inline]
```

Inform a garbage collector that a region of memory need not be traced.

#### `declare_reachable()`

```
void std::declare_reachable (  
    void * ) [inline]
```

Inform a garbage collector that an object is still in use.

#### `get_pointer_safety()`

```
pointer_safety std::get_pointer_safety () [inline], [noexcept]
```

The type of pointer safety supported by the implementation.

#### `undeclare_no_pointers()`

```
void std::undeclare_no_pointers (  
    char * ,  
    size_t ) [inline]
```

Unregister a range previously registered with `declare_no_pointers`.

#### `undeclare_reachable()`

```
template<typename _Tp>  
_Tp * std::undeclare_reachable (  
    _Tp * __p) [inline]
```

Unregister an object previously registered with `declare_reachable`.

### 3.15.5.6 Polymorphic memory resources

Collaboration diagram for Polymorphic memory resources:



#### Files

- file [memory\\_resource](#)

#### Classes

- struct [std::allocator\\_traits< pmr::polymorphic\\_allocator< \\_Tp > >](#)
- class [std::pmr::memory\\_resource](#)
- class [std::pmr::monotonic\\_buffer\\_resource](#)
- class [std::pmr::polymorphic\\_allocator< \\_Tp >](#)
- struct [std::pmr::pool\\_options](#)
- class [std::pmr::unsynchronized\\_pool\\_resource](#)

#### Functions

- [memory\\_resource \\* std::pmr::new\\_delete\\_resource \(\) noexcept](#)

#### 3.15.5.6.1 Detailed Description

Since

C++17

Memory resources are classes that implement the `std::pmr::memory_resource` interface for allocating and deallocating memory. Unlike traditional C++ allocators, memory resources are not value types and are used via pointers to the abstract base class. They are only responsible for allocating and deallocating, not for construction and destruction of objects. As a result, memory resources just allocate raw memory as type `void*` and are not templates that allocate/deallocate and construct/destroy a specific type.

The class template `std::pmr::polymorphic_allocator` is an allocator that uses a memory resource for its allocations.

#### 3.15.5.6.2 Function Documentation

##### `new_delete_resource()`

`memory_resource * std::pmr::new_delete_resource () [nodiscard], [noexcept]`

A `pmr::memory_resource` that uses `new` to allocate memory.

Since

C++17

### 3.15.6 Metaprogramming

Collaboration diagram for Metaprogramming:



#### Classes

- struct `std::tr2::__reflection_typelist< _Elements >`
- struct `std::tr2::__reflection_typelist< _First, _Rest... >`
- struct `std::tr2::__reflection_typelist<>`
- struct `std::add_const< _Tp >`
- struct `std::add_cv< _Tp >`
- struct `std::add_lvalue_reference< _Tp >`
- struct `std::add_pointer< _Tp >`
- struct `std::add_rvalue_reference< _Tp >`
- struct `std::add_volatile< _Tp >`
- struct `std::aligned_storage< _Len, _Align >`
- struct `std::aligned_union< _Len, _Types >`
- struct `std::alignment_of< _Tp >`
- struct `std::tr2::bases< _Tp >`
- struct `std::common_type< _Tp >`
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
- struct `std::decay< _Tp >`
- struct `std::tr2::direct_bases< _Tp >`
- struct `std::enable_if< bool, _Tp >`
- struct `std::extent< typename, _Uint >`
- struct `std::has_virtual_destructor< _Tp >`
- struct `std::integral_constant< _Tp, __v >`
- struct `std::is_abstract< _Tp >`
- struct `std::is_arithmetic< _Tp >`
- struct `std::is_array< _Tp >`
- struct `std::is_assignable< _Tp, _Up >`
- struct `std::is_base_of< _Base, _Derived >`
- struct `std::is_class< _Tp >`
- struct `std::is_compound< _Tp >`
- struct `std::is_const< _Tp >`
- struct `std::is_constructible< _Tp, _Args >`
- struct `std::is_copy_assignable< _Tp >`
- struct `std::is_copy_constructible< _Tp >`
- struct `std::is_default_constructible< _Tp >`
- struct `std::is_destructible< _Tp >`
- struct `std::is_empty< _Tp >`
- struct `std::is_enum< _Tp >`

- struct `std::is_floating_point< _Tp >`
- struct `std::is_function< _Tp >`
- struct `std::is_fundamental< _Tp >`
- struct `std::is_integral< _Tp >`
- struct `std::is_layout_compatible< _Tp, _Up >`
- struct `std::is_literal_type< _Tp >`
- struct `std::is_lvalue_reference< typename >`
- struct `std::is_member_function_pointer< _Tp >`
- struct `std::is_member_object_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_move_assignable< _Tp >`
- struct `std::is_move_constructible< _Tp >`
- struct `std::is_nothrow_assignable< _Tp, _Up >`
- struct `std::is_nothrow_constructible< _Tp, _Args >`
- struct `std::is_nothrow_copy_assignable< _Tp >`
- struct `std::is_nothrow_copy_constructible< _Tp >`
- struct `std::is_nothrow_default_constructible< _Tp >`
- struct `std::is_nothrow_destructible< _Tp >`
- struct `std::is_nothrow_move_assignable< _Tp >`
- struct `std::is_nothrow_move_constructible< _Tp >`
- struct `std::is_object< _Tp >`
- struct `std::is_pod< _Tp >`
- struct `std::is_pointer< _Tp >`
- struct `std::is_pointer_interconvertible_base_of< _Base, _Derived >`
- struct `std::is_polymorphic< _Tp >`
- struct `std::is_reference< _Tp >`
- struct `std::is_rvalue_reference< typename >`
- struct `std::is_same< _Tp, _Up >`
- struct `std::is_scalar< _Tp >`
- struct `std::is_signed< _Tp >`
- struct `std::is_standard_layout< _Tp >`
- struct `std::is_trivial< _Tp >`
- struct `std::is_trivially_assignable< _Tp, _Up >`
- struct `std::is_trivially_constructible< _Tp, _Args >`
- struct `std::is_trivially_copy_assignable< _Tp >`
- struct `std::is_trivially_copy_constructible< _Tp >`
- struct `std::is_trivially_copyable< _Tp >`
- struct `std::is_trivially_default_constructible< _Tp >`
- struct `std::is_trivially_destructible< _Tp >`
- struct `std::is_trivially_move_assignable< _Tp >`
- struct `std::is_trivially_move_constructible< _Tp >`
- struct `std::is_union< _Tp >`
- struct `std::is_unsigned< _Tp >`
- struct `std::is_void< _Tp >`
- struct `std::is_volatile< _Tp >`
- struct `std::make_signed< _Tp >`
- struct `std::make_unsigned< _Tp >`
- struct `std::rank< _Tp >`
- struct `std::remove_all_extents< _Tp >`
- struct `std::remove_const< _Tp >`
- struct `std::remove_cv< _Tp >`

- struct [std::remove\\_extent<\\_Tp>](#)
- struct [std::remove\\_pointer<\\_Tp>](#)
- struct [std::remove\\_reference<\\_Tp>](#)
- struct [std::remove\\_volatile<\\_Tp>](#)
- struct [std::result\\_of<\\_Signature>](#)
- struct [std::underlying\\_type<\\_Tp>](#)

## Typedefs

- template<bool \_Cond, typename \_If, typename \_Else>  
using **std::\_\_conditional\_t**
- template<bool \_Cond, typename \_Tp = void>  
using **std::\_\_enable\_if\_t**
- template<typename \_ToElementType, typename \_FromElementType>  
using **std::\_\_is\_array\_convertible**
- using **std::Build\_index\_tuple<\_Num>::\_\_type**
- template<typename, size\_t... \_Indices>  
using **std::Build\_index\_tuple<\_Num>::\_\_IdxTuple**
- template<typename \_Tp>  
using [std::add\\_lvalue\\_reference\\_t](#)
- template<typename \_Tp>  
using [std::add\\_pointer\\_t](#)
- template<typename \_Tp>  
using [std::add\\_rvalue\\_reference\\_t](#)
- template<size\_t \_Len, size\_t \_Align = \_\_aligned\_storage\_default\_alignment(\_Len)>  
using [std::aligned\\_storage\\_t](#)
- template<size\_t \_Len, typename... \_Types>  
using **std::aligned\_union\_t**
- template<typename... \_Tp>  
using [std::common\\_type\\_t](#)
- template<bool \_Cond, typename \_Iftrue, typename \_Iffalse>  
using [std::conditional\\_t](#)
- template<typename \_Tp>  
using [std::decay\\_t](#)
- template<bool \_Cond, typename \_Tp = void>  
using [std::enable\\_if\\_t](#)
- using [std::false\\_type](#)
- template<typename \_Tp>  
using [std::make\\_signed\\_t](#)
- template<typename \_Tp>  
using [std::make\\_unsigned\\_t](#)
- template<typename \_Tp>  
using [std::remove\\_all\\_extents\\_t](#)
- template<typename \_Tp>  
using [std::remove\\_extent\\_t](#)
- template<typename \_Tp>  
using [std::remove\\_pointer\\_t](#)
- template<typename \_Tp>  
using [std::remove\\_reference\\_t](#)
- template<typename \_Tp>  
using [std::result\\_of\\_t](#)
- using [std::true\\_type](#)

- `template<typename _Tp, typename>`  
    using `std::__conditional< bool >::type`
- `template<typename, typename _Up>`  
    using `std::__conditional< false >::type`
- using `std::__is_implicitly_default_constructible_impl< _Tp >::type`
- using `std::add_const< _Tp >::type`
- using `std::add_cv< _Tp >::type`
- using `std::add_lvalue_reference< _Tp >::type`
- using `std::add_pointer< _Tp >::type`
- using `std::add_rvalue_reference< _Tp >::type`
- using `std::add_volatile< _Tp >::type`
- using `std::aligned_union< _Len, _Types >::type`
- using `std::common_reference< _Tp0 >::type`
- using `std::conditional< _Cond, _Iftrue, _Iffalse >::type`
- using `std::conditional< false, _Iftrue, _Iffalse >::type`
- using `std::decay< _Tp >::type`
- using `std::decay< _Tp & >::type`
- using `std::decay< _Tp && >::type`
- using `std::enable_if< true, _Tp >::type`
- using `std::integral_constant< _Tp, __v >::type`
- using `std::make_signed< _Tp >::type`
- using `std::make_unsigned< _Tp >::type`
- using `std::remove_all_extents< _Tp >::type`
- using `std::remove_const< _Tp >::type`
- using `std::remove_const< _Tp const >::type`
- using `std::remove_cv< _Tp >::type`
- using `std::remove_extent< _Tp >::type`
- using `std::remove_pointer< _Tp >::type`
- using `std::remove_reference< _Tp >::type`
- using `std::remove_volatile< _Tp >::type`
- using `std::remove_volatile< _Tp volatile >::type`
- `template<typename _Tp>`  
    using `std::underlying_type_t`
- using `std::integral_constant< _Tp, __v >::value_type`

## Functions

- `template<typename _Tp>`  
    static void `std::__do_is_implicitly_default_constructible_impl::__helper` (const \_Tp &)
- static `false_type` `std::__do_is_implicitly_default_constructible_impl::__test` (...)
- `template<typename _Tp>`  
    static `true_type` `std::__do_is_implicitly_default_constructible_impl::__test` (const \_Tp &, decltype(\_\_helper< const \_Tp & >({})) \* = 0)
- constexpr `std::integral_constant< _Tp, __v >::operator value_type` () const noexcept
- `template<typename _Tp>`  
    constexpr `_Require< __not< __is_tuple_like< _Tp >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > std::swap` (\_Tp &, \_Tp &) noexcept(\_\_and< is\_nothrow\_move\_constructible< \_Tp >, is\_nothrow\_move\_assignable< \_Tp > >::value)
- `template<typename _Tp, size_t _Nm>`  
    constexpr `__enable_if_t< __is_swappable< _Tp >::value > std::swap` (\_Tp(&\_\_a)[\_Nm], \_Tp(&\_\_b)[\_Nm]) noexcept(\_\_is\_nothrow\_swappable< \_Tp >::value)



## Variables

- unsigned char **std::aligned\_storage**< \_Len, \_Align >::type::\_\_data [\_Len]
- template<typename \_Tp, typename... \_Args>  
constexpr bool **std::\_\_is\_nothrow\_new\_constructible**
- static const size\_t **std::\_\_strictest\_alignment**< \_Types >::\_\_S\_alignment
- static const size\_t **std::\_\_strictest\_alignment**< \_Tp, \_Types... >::\_\_S\_alignment
- static const size\_t **std::\_\_strictest\_alignment**< \_Types >::\_\_S\_size
- static const size\_t **std::\_\_strictest\_alignment**< \_Tp, \_Types... >::\_\_S\_size
- template<size\_t \_Len, typename... \_Types>  
const size\_t **std::aligned\_union**< \_Len, \_Types... >::alignment\_value
- static const size\_t **std::aligned\_union**< \_Len, \_Types >::alignment\_value
- static constexpr \_Tp **std::integral\_constant**< \_Tp, \_\_v >::value
- template<typename... \_Tp>  
using **std::common\_reference\_t**
- template<typename \_Tp, typename \_Up>  
constexpr bool **std::is\_layout\_compatible\_v**
- template<typename \_Base, typename \_Derived>  
constexpr bool **std::is\_pointer\_interconvertible\_base\_of\_v**
- template<typename \_S1, typename \_S2, typename \_M1, typename \_M2>  
constexpr bool **std::is\_corresponding\_member** (\_M1 \_S1::\*\_\_m1, \_M2 \_S2::\*\_\_m2) noexcept
- template<typename \_Tp, typename \_Mem>  
constexpr bool **std::is\_pointer\_interconvertible\_with\_class** (\_Mem \_Tp::\*\_\_mp) noexcept

### 3.15.6.1 Detailed Description

Template utilities for compile-time introspection and modification, including type classification traits, type property inspection traits and type transformation traits.

Since

C++11

### 3.15.6.2 Typedef Documentation

#### **add\_lvalue\_reference\_t**

```
template<typename _Tp>
using std::add_lvalue_reference_t
Alias template for add_lvalue_reference.
```

#### **add\_pointer\_t**

```
template<typename _Tp>
using std::add_pointer_t
Alias template for add_pointer.
```

#### **add\_rvalue\_reference\_t**

```
template<typename _Tp>
using std::add_rvalue_reference_t
Alias template for add_rvalue_reference.
```

**aligned\_storage\_t**

```
template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
using std::aligned_storage_t
Alias template for aligned_storage.
```

**common\_reference\_t**

```
template<typename... _Tp>
using std::common_reference_t
```

Since

C++20

**common\_type\_t**

```
template<typename... _Tp>
using std::common_type_t
Alias template for common_type.
```

**conditional\_t**

```
template<bool _Cond, typename _Iftrue, typename _Iffalse>
using std::conditional_t
Alias template for conditional.
```

**decay\_t**

```
template<typename _Tp>
using std::decay_t
Alias template for decay.
```

**enable\_if\_t**

```
template<bool _Cond, typename _Tp = void>
using std::enable_if_t
Alias template for enable_if.
```

**false\_type**

```
using std::false_type
The type used as a compile-time boolean with false value.
```

**make\_signed\_t**

```
template<typename _Tp>
using std::make_signed_t
Alias template for make_signed.
```

**make\_unsigned\_t**

```
template<typename _Tp>
using std::make_unsigned_t
Alias template for make_unsigned.
```

**remove\_all\_extents\_t**

```
template<typename _Tp>
using std::remove_all_extents_t
Alias template for remove_all_extents.
```

**remove\_extent\_t**

```
template<typename _Tp>
using std::remove_extent_t
Alias template for remove_extent.
```

**remove\_pointer\_t**

```
template<typename _Tp>
using std::remove_pointer_t
Alias template for remove_pointer.
```

**remove\_reference\_t**

```
template<typename _Tp>
using std::remove_reference_t
Alias template for remove_reference.
```

**result\_of\_t**

```
template<typename _Tp>
using std::result_of_t
Alias template for result_of.
```

**true\_type**

```
using std::true_type
The type used as a compile-time boolean with true value.
```

**type**

```
template<size_t _Len, typename... _Types>
using std::aligned_union< _Len, _Types >::type
The storage.
```

**underlying\_type\_t**

```
template<typename _Tp>
using std::underlying_type_t
Alias template for underlying_type.
```

**3.15.6.3 Function Documentation****is\_corresponding\_member()**

```
template<typename _S1, typename _S2, typename _M1, typename _M2>
bool std::is_corresponding_member (
    _M1 _S1::* __m1,
    _M2 _S2::* __m2) [constexpr], [noexcept]
```

Since

C++20

### is\_pointer\_interconvertible\_with\_class()

```
template<typename _Tp, typename _Mem>
bool std::is_pointer_interconvertible_with_class (
    _Mem __mp) [constexpr], [noexcept]
```

True if `__mp` points to the first member of a standard-layout type.

Returns

true if `s.*__mp` is pointer-interconvertible with `s`

Since

C++20

### swap() [1/2]

```
template<typename _Tp>
_Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable<
_Tp > > std::swap (
    _Tp & __a,
    _Tp & __b) [inline], [constexpr], [noexcept]
```

Swaps two values.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

Nothing.

### swap() [2/2]

```
template<typename _Tp, size_t _Nm>
__enable_if_t< __is_swappable< _Tp >::value > std::swap (
    _Tp(&) __a[_Nm],
    _Tp(&) __b[_Nm]) [inline], [constexpr], [noexcept]
```

Swap the contents of two arrays.

#### 3.15.6.4 Variable Documentation

##### alignment\_value

```
template<size_t _Len, typename... _Types>
const size_t std::aligned_union< _Len, _Types >::alignment_value [static]
```

The value of the strictest alignment of `_Types`.

**is\_layout\_compatible\_v**

```
template<typename _Tp, typename _Up>
bool std::is_layout_compatible_v [constexpr]
```

Since

C++20

**is\_pointer\_interconvertible\_base\_of\_v**

```
template<typename _Base, typename _Derived>
bool std::is_pointer_interconvertible_base_of_v [constexpr]
```

Since

C++20

**3.15.7 Rational Arithmetic**

Collaboration diagram for Rational Arithmetic:

**Files**

- file [ratio](#)

**Classes**

- struct [std::ratio<\\_Num, \\_Den>](#)
- struct [std::ratio\\_equal<\\_R1, \\_R2>](#)
- struct [std::ratio\\_greater<\\_R1, \\_R2>](#)
- struct [std::ratio\\_greater\\_equal<\\_R1, \\_R2>](#)
- struct [std::ratio\\_less<\\_R1, \\_R2>](#)
- struct [std::ratio\\_less\\_equal<\\_R1, \\_R2>](#)
- struct [std::ratio\\_not\\_equal<\\_R1, \\_R2>](#)

**Typedefs**

- using **std::atto**
- using **std::centi**
- using **std::deca**
- using **std::deci**
- using **std::exa**
- using **std::femto**
- using **std::giga**

- using **std::hecto**
- using **std::kilo**
- using **std::mega**
- using **std::micro**
- using **std::milli**
- using **std::nano**
- using **std::peta**
- using **std::pico**
- template<typename \_R1, typename \_R2>  
using **std::ratio\_add**
- template<typename \_R1, typename \_R2>  
using **std::ratio\_divide**
- template<typename \_R1, typename \_R2>  
using **std::ratio\_multiply**
- template<typename \_R1, typename \_R2>  
using **std::ratio\_subtract**
- using **std::tera**

### Variables

- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_equal\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_greater\_equal\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_greater\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_less\_equal\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_less\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_not\_equal\_v**

#### 3.15.7.1 Detailed Description

Compile time representation of finite rational numbers.

#### 3.15.7.2 Typedef Documentation

##### **ratio\_add**

```
template<typename _R1, typename _R2>  
using std::ratio_add  
ratio_add
```

##### **ratio\_divide**

```
template<typename _R1, typename _R2>  
using std::ratio_divide  
ratio_divide
```

**ratio\_multiply**

```
template<typename _R1, typename _R2>
using std::ratio_multiply
ratio_multiply
```

**ratio\_subtract**

```
template<typename _R1, typename _R2>
using std::ratio_subtract
ratio_subtract
```

**3.15.8 Time**

Collaboration diagram for Time:

**Files**

- file [chrono](#)

**Namespaces**

- namespace [std::chrono](#)
- namespace [std::literals::chrono\\_literals](#)

**Classes**

- struct [std::common\\_type< chrono::duration< \\_Rep, \\_Period > >](#)
- struct [std::common\\_type< chrono::duration< \\_Rep, \\_Period >, chrono::duration< \\_Rep, \\_Period > >](#)
- struct [std::common\\_type< chrono::duration< \\_Rep1, \\_Period1 >, chrono::duration< \\_Rep2, \\_Period2 > >](#)
- struct [std::common\\_type< chrono::time\\_point< \\_Clock, \\_Duration > >](#)
- struct [std::common\\_type< chrono::time\\_point< \\_Clock, \\_Duration >, chrono::time\\_point< \\_Clock, \\_Duration > >](#)
- struct [std::common\\_type< chrono::time\\_point< \\_Clock, \\_Duration1 >, chrono::time\\_point< \\_Clock, \\_Duration2 > >](#)
- class [std::chrono::duration< \\_Rep, \\_Period >](#)
- struct [std::chrono::duration\\_values< \\_Rep >](#)
- class [std::chrono::gps\\_clock](#)
- class [std::chrono::hh\\_mm\\_ss< \\_Duration >](#)
- struct [std::chrono::steady\\_clock](#)
- struct [std::chrono::system\\_clock](#)
- class [std::chrono::tai\\_clock](#)
- class [std::chrono::time\\_point< \\_Clock, \\_Dur >](#)
- struct [std::chrono::treat\\_as\\_floating\\_point< \\_Rep >](#)
- class [std::chrono::tzdb\\_list](#)
- class [std::chrono::utc\\_clock](#)

## Typedefs

- using `std::chrono::days`
- using `std::chrono::file_clock`
- template<typename \_Duration>  
using `std::chrono::file_time`
- using `std::chrono::gps_seconds`
- template<typename \_Duration>  
using `std::chrono::gps_time`
- using `std::chrono::high_resolution_clock`
- using `std::chrono::hours`
- using `std::chrono::local_days`
- using `std::chrono::local_seconds`
- template<typename \_Duration>  
using `std::chrono::local_time`
- using `std::chrono::microseconds`
- using `std::chrono::milliseconds`
- using `std::chrono::minutes`
- using `std::chrono::months`
- using `std::chrono::nanoseconds`
- using `std::chrono::seconds`
- using `std::chrono::sys_days`
- using `std::chrono::sys_seconds`
- template<typename \_Duration>  
using `std::chrono::sys_time`
- using `std::chrono::tai_seconds`
- template<typename \_Duration>  
using `std::chrono::tai_time`
- using `std::chrono::utc_seconds`
- template<typename \_Duration>  
using `std::chrono::utc_time`
- using `std::chrono::weeks`
- using `std::chrono::years`
- using `std::chrono::zoned_seconds`

## Enumerations

- enum class `choose` { `earliest` , `latest` }

## Functions

- constexpr `std::chrono::year_month_day::year_month_day` (const year\_month\_day\_last &\_\_ymdl) noexcept
- template<typename \_Duration>  
void `std::chrono::__throw_bad_local_time` (const `local_time`< \_Duration > &\_\_tp, const local\_info &\_\_i)
- template<typename \_Rep, typename \_Period>  
constexpr `enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > >` `std::chrono::abs` (`duration`< \_Rep, \_Period > \_\_d)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr `__enable_if_is_duration< _ToDur >` `std::chrono::ceil` (const `duration`< \_Rep, \_Period > &\_\_d)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr `enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > >` `std::chrono::ceil` (const `time_point`< \_Clock, \_Dur > &\_\_tp)



- `template<typename _DestClock, typename _SourceClock, typename _Duration>`  
`requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>`  
`auto std::chrono::clock_cast (const time_point< _SourceClock, _Duration > &__t)`
- `const time_zone * std::chrono::current_zone ()`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr __enable_if_is_duration< _ToDur > std::chrono::duration_cast (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr __enable_if_is_duration< _ToDur > std::chrono::floor (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (const time_point< _Clock, _Dur > &__tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, day & __d, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Rep, typename _Period, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, duration< _Rep, _Period > & __d, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, file_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, gps_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, local_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, month & __m, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, month_day & __md, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, sys_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, tai_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, utc_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`

- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, weekday &__wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, year &__y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, year_month &__ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, year_month_day &__ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes`  
`*__offset=nullptr)`
- `template<typename _Duration>`  
`static utc_time< common_type_t< _Duration, seconds > > std::chrono::utc_clock::from_sys (const`  
`sys_time< _Duration > &__t)`
- `template<typename _Duration>`  
`leap_second_info std::chrono::get_leap_second_info (const utc_time< _Duration > &__ut)`
- `const tzdb & std::chrono::get_tzdb ()`
- `tzdb_list & std::chrono::get_tzdb_list ()`
- `constexpr bool std::chrono::is_am (const hours &__h) noexcept`
- `constexpr bool std::chrono::is_pm (const hours &__h) noexcept`
- `template<typename _Duration>`  
`__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (local_time< _Duration > __time, const`  
`string *__abbrev=nullptr, const seconds *__offset_sec=nullptr)`
- `const time_zone * std::chrono::locate_zone (string_view __tz_name)`
- `constexpr hours std::chrono::make12 (const hours &__h) noexcept`
- `constexpr hours std::chrono::make24 (const hours &__h, bool __is_pm) noexcept`
- `constexpr bool std::chrono::year_month_day::ok () const noexcept`
- `constexpr chrono::day std::literals::chrono_literals::operator""d (unsigned long long __d) noexcept`
- `template<char... _Digits>`  
`constexpr chrono::hours std::literals::chrono_literals::operator""h ()`
- `constexpr chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (long dou-`  
`ble __hours)`
- `template<char... _Digits>`  
`constexpr chrono::minutes std::literals::chrono_literals::operator""min ()`
- `constexpr chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (long dou-`  
`ble __mins)`
- `template<char... _Digits>`  
`constexpr chrono::milliseconds std::literals::chrono_literals::operator""ms ()`
- `constexpr chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (long double __`  
`msecs)`
- `template<char... _Digits>`  
`constexpr chrono::nanoseconds std::literals::chrono_literals::operator""ns ()`
- `constexpr chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (long double __nsecs)`
- `template<char... _Digits>`  
`constexpr chrono::seconds std::literals::chrono_literals::operator""s ()`
- `constexpr chrono::duration< long double > std::literals::chrono_literals::operator""s (long double __secs)`
- `template<char... _Digits>`  
`constexpr chrono::microseconds std::literals::chrono_literals::operator""us ()`

- constexpr `chrono::duration< long double, micro > std::literals::chrono_literals::operator""us` (long double \_\_↵ usecs)
- constexpr `chrono::year std::literals::chrono_literals::operator""y` (unsigned long long \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const day &__d)`
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const file_time< _Duration > &__t)`
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const gps_time< _Duration > &__t)`
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const hh_mm_ss< _Duration > &__hms)`
- template<typename \_CharT, typename \_Traits>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info &__li)`
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
requires requires(const `sys_time<_Duration>& __st`) { \_\_os << \_\_st; }  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_time< _Duration > &__lt)`
- template<typename \_CharT, typename \_Traits>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month &__m)`
- template<typename \_CharT, typename \_Traits>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_day &__md)`
- template<typename \_CharT, typename \_Traits>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_day_last &__mdl)`
- template<typename \_CharT, typename \_Traits>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_weekday &__mwd)`
- template<typename \_CharT, typename \_Traits>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_weekday_last &__mwdl)`
- template<typename \_CharT, typename \_Traits>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_days &__dp)`
- template<typename \_CharT, typename \_Traits>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_info &__i)`
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
requires (!treat\_as\_floating\_point\_v<typename \_Duration::rep>) && ratio\_less\_v<typename \_Duration::period, days::period>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_time< _Duration > &__tp)`
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const tai_time< _Duration > &__t)`
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const utc_time< _Duration > &__t)`

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const weekday &__wd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const weekday_indexed &__wdi)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const weekday_last &__wdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year &__y)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month &__ym)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_day &__ymd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_day_last &__ymdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_weekday &__ymwd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_weekday_last &__ymwdl)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const zoned_time< _Duration, _TimeZonePtr > &__t)`
- `template<typename _CharT, typename _Traits, typename _Rep, typename _Period>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const duration< _Rep, _Period > &__d)`
- `template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>`  
`bool std::chrono::operator== (const zoned_time< _Dur1, _TZPtr1 > &__x, const zoned_time< _Dur2, _TZPtr2`  
`> &__y)`
- `constexpr weekday_last std::chrono::weekday::operator[] (last_spec) const noexcept`
- `constexpr weekday_indexed std::chrono::weekday::operator[] (unsigned __index) const noexcept`
- `template<typename _CharT, __detail::__parsable< _CharT > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, basic_string< _CharT, _Traits, _Alloc > & __↵`  
`_abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, basic_string< _CharT, _Traits, _Alloc > & __↵`  
`_abbrev, minutes & __offset)`
- `template<typename _CharT, typename _Traits = char_traits< _CharT>, typename _StrT = basic_string< _CharT, _Traits>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, minutes & __offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>`  
`auto std::chrono::parse (const basic_string< _CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp)`

- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↳  
parsable< _CharT, _Traits, _StrT > _Parsable>  
auto std::chrono::parse (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string<  
_CharT, _Traits, _Alloc > &__abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↳  
parsable< _CharT, _Traits, _StrT, minutes > _Parsable>  
auto std::chrono::parse (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string<  
_CharT, _Traits, _Alloc > &__abbrev, minutes &__offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__↳  
parsable< _CharT, _Traits, _StrT, minutes > _Parsable>  
auto std::chrono::parse (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, minutes &__↳  
__offset)`
- `const tzdb & std::chrono::reload_tzdb ()`
- `string std::chrono::remote_version ()`
- `template<typename _ToDur, typename _Rep, typename _Period>  
constexpr enable_if_t< __and< __is_duration< _ToDur >, __not< treat_as_floating_point< typename _To↳  
Dur::rep > >::value, _ToDur > std::chrono::round (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur>  
constexpr enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >,   
time_point< _Clock, _ToDur > > std::chrono::round (const time_point< _Clock, _Dur > &__tp)`
- `template<typename _ToDur, typename _Clock, typename _Dur>  
constexpr __enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono::time_point_cast  
(const time_point< _Clock, _Dur > &__t)`
- `std::chrono::zoned_time () -> zoned_time< seconds >`
- `template<typename _TimeZonePtrOrName>  
std::chrono::zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation<  
_TimeZonePtrOrName > >`
- `template<typename _TimeZonePtrOrName, typename _Duration>  
std::chrono::zoned_time (_TimeZonePtrOrName &&, local_time< _Duration >, choose=choose::earliest) ->  
zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName  
> >`
- `template<typename _TimeZonePtrOrName, typename _Duration>  
std::chrono::zoned_time (_TimeZonePtrOrName &&, sys_time< _Duration >) -> zoned_time< common_type_t<  
_Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >`
- `template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>  
std::chrono::zoned_time (_TimeZonePtrOrName &&, zoned_time< _Duration, _TimeZonePtr2 >, choose=choose↳  
::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _Time↳  
ZonePtrOrName > >`
- `template<typename _Duration>  
std::chrono::zoned_time (sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >  
>`

## Variables

- `template<typename _Tp, typename _Clock>  
constexpr bool std::chrono::__is_time_point_for_v`
- `template<typename _Clock, typename _Duration>  
constexpr bool std::chrono::__is_time_point_for_v< time_point< _Clock, _Duration >, _Clock >`
- `constexpr month std::chrono::April`
- `constexpr month std::chrono::August`
- `constexpr month std::chrono::December`
- `constexpr month std::chrono::February`
- `constexpr weekday std::chrono::Friday`

- `template<typename _Tp, typename = void>`  
`constexpr bool std::chrono::is_clock_v`
- `template<typename _Tp>`  
`constexpr bool std::chrono::is_clock_v< _Tp, void_t< typename _Tp::rep, typename _Tp::period, type-`  
`name _Tp::duration, typename _Tp::time_point::duration, decltype(_Tp::is_steady), decltype(_Tp::now())>>`
- `template<> constexpr bool std::chrono::is_clock_v< file_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< gps_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< steady_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< system_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< tai_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< utc_clock >`
- `constexpr month std::chrono::January`
- `constexpr month std::chrono::July`
- `constexpr month std::chrono::June`
- `constexpr last_spec std::chrono::last`
- `constexpr month std::chrono::March`
- `constexpr month std::chrono::May`
- `constexpr weekday std::chrono::Monday`
- `constexpr month std::chrono::November`
- `constexpr month std::chrono::October`
- `constexpr weekday std::chrono::Saturday`
- `constexpr month std::chrono::September`
- `constexpr weekday std::chrono::Sunday`
- `constexpr weekday std::chrono::Thursday`
- `template<typename _Rep>`  
`constexpr bool std::chrono::treat_as_floating_point_v`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< double >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< float >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< int >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long double >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long long >`
- `constexpr weekday std::chrono::Tuesday`
- `constexpr weekday std::chrono::Wednesday`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator-`  
`(const duration< _Rep1, _Period1 > &_lhs, const duration< _Rep2, _Period2 > &_rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+`  
`(const duration< _Rep1, _Period1 > &_lhs, const duration< _Rep2, _Period2 > &_rhs)`
- `template<typename _Rep1, typename _Rep2, typename _Period>`  
`constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (const _Rep1 &_s,`  
`const duration< _Rep2, _Period > &_d)`
- `template<typename _Rep1, typename _Period, typename _Rep2>`  
`constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator/ (const duration< _Rep1, _Period > &_d, const _Rep2 &_s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 >`  
`&_lhs, const duration< _Rep2, _Period2 > &_rhs)`



- `template<typename _Rep1, typename _Period, typename _Rep2>`  
`constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period >`  
`std::chrono::operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator%`  
`(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2>`  
`constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, _`  
`Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`requires three_way_comparable<common_type_t<_Rep1, _Rep2>>`  
`constexpr auto std::chrono::operator<=> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>`  
`constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type >`  
`std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__`  
`__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >`  
`std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__`  
`__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 > &__`  
`__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >`  
`operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>`  
`constexpr auto std::chrono::operator<=> (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _`  
`Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator< (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator<= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _`  
`Clock, _Dur2 > &__rhs)`

- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator> (const time\_point< _Clock, _Dur1 > &__lhs, const time\_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator>= (const time\_point< _Clock, _Dur1 > &__lhs, const time\_point< _Clock,`  
`_Dur2 > &__rhs)`

### 3.15.8.1 Detailed Description

Classes and functions for time.

Since

C++11

### 3.15.8.2 Typedef Documentation

#### **days**

using [std::chrono::days](#)  
days

#### **high\_resolution\_clock**

using [std::chrono::high\\_resolution\\_clock](#)

Highest-resolution clock.

This is the clock "with the shortest tick period." Alias to `std::system_clock` until higher-than-nanosecond definitions become feasible.

#### **hours**

using [std::chrono::hours](#)  
hours

#### **microseconds**

using [std::chrono::microseconds](#)  
microseconds

#### **milliseconds**

using [std::chrono::milliseconds](#)  
milliseconds

#### **minutes**

using [std::chrono::minutes](#)  
minutes

#### **months**

using [std::chrono::months](#)  
months

#### **nanoseconds**

using [std::chrono::nanoseconds](#)  
nanoseconds



**seconds**

```
using std::chrono::seconds
seconds
```

**weeks**

```
using std::chrono::weeks
weeks
```

**years**

```
using std::chrono::years
years
```

**3.15.8.3 Function Documentation****abs()**

```
template<typename _Rep, typename _Period>
enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > std::chrono::abs (
    duration< _Rep, _Period > __d) [nodiscard], [constexpr]
```

The absolute (non-negative) value of a duration.

**Parameters**

$\leftrightarrow$ _d	A duration with a signed rep type.
-------------------------	------------------------------------

**Returns**

A duration of the same type as the argument, with value  $|d|$ .

**Since**

C++17

**ceil() [1/2]**

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::ceil (
    const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type `ToDur` and round up.

If the duration cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

**Template Parameters**

<code>_ToDur</code>	The result type must be a duration.
---------------------	-------------------------------------

**Parameters**

$\leftrightarrow$ _d	A duration.
-------------------------	-------------

**Returns**

The value of `__d` converted to type `_ToDur`.

**Since**

C++17

References [duration\\_cast\(\)](#).

Referenced by [ceil\(\)](#).

**ceil()** [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::ceil (
    const time_point< _Clock, _Dur > & __tp) [nodiscard], [constexpr]
```

Convert a `time_point` to type `ToDur` and round up.

The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

**Template Parameters**

<code>_ToDur</code>	The duration type to use for the result.
---------------------	--

**Parameters**

<code>↔</code>	A time point.
<code>↔</code>	
<code>↔</code>	
<code>↔</code>	
<code>t</code>	

**Returns**

The value of `__d` converted to type `_ToDur`.

**Since**

C++17

References [ceil\(\)](#).

**clock\_cast()**

```
template<typename _DestClock, typename _SourceClock, typename _Duration>
requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs↔
_sys<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock,
_Duration> || __detail::__clock_convs_sys_utc<_DestClock, _SourceClock, _Duration> || __detail::__↔
_clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
auto std::chrono::clock_cast (
    const time_point< _SourceClock, _Duration > & __t) [inline], [nodiscard]
```

Convert a time point to a different clock.

**duration\_cast()**

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::duration_cast (
    const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type `ToDur`.

If the duration cannot be represented accurately in the result type, returns the result of integer truncation (i.e., rounded towards zero).

**Template Parameters**

<code>_ToDur</code>	The result type must be a duration.
---------------------	-------------------------------------

**Parameters**

<code>__d</code>	A duration.
------------------	-------------

**Returns**

The value of `__d` converted to type `_ToDur`.

**Since**

C++11

Referenced by [ceil\(\)](#), [floor\(\)](#), [std::this\\_thread::sleep\\_for\(\)](#), and [time\\_point\\_cast\(\)](#).

**floor()** [1/2]

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::floor (
    const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type `ToDur` and round down.

If the duration cannot be represented exactly in the result type, returns the closest value that is less than the argument.

**Template Parameters**

<code>_ToDur</code>	The result type must be a duration.
---------------------	-------------------------------------

**Parameters**

<code>__d</code>	A duration.
------------------	-------------

**Returns**

The value of `__d` converted to type `_ToDur`.

**Since**

C++17

References [duration\\_cast\(\)](#).

Referenced by [floor\(\)](#), and [round\(\)](#).

**floor()** [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (
    const time_point< _Clock, _Dur > & __tp) [nodiscard], [constexpr]
```

Convert a `time_point` to type `ToDur` and round down.

The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value that is less than the argument.

**Template Parameters**

<code>_ToDur</code>	The <code>duration</code> type to use for the result.
---------------------	---

**Parameters**

<code>↵</code>	A time point.
<code>↵</code>	
<code>↵</code>	
<code>↵</code>	
<code>t</code>	

**Returns**

The value of `__d` converted to type `_ToDur`.

**Since**

C++17

References [floor\(\)](#).

**local\_time\_format()**

```
template<typename _Duration>
__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (
    local_time< _Duration > __time,
    const string * __abbrev = nullptr,
    const seconds * __offset_sec = nullptr) [inline]
```

Return an object that associates timezone info with a local time.

A `chrono::local_time` object has no timezone associated with it. This function creates an object that allows formatting a `local_time` as though it refers to a timezone with the given abbreviated name and offset from UTC.

**Since**

C++20

**operator""d()**

```
chrono::day std::literals::chrono_literals::operator""d (
    unsigned long long __d) [constexpr], [noexcept]
```

Literal suffix for creating `chrono::day` objects.

**Since**

C++20

**operator""h() [1/2]**

```
template<char... _Digits>
chrono::hours std::literals::chrono_literals::operator""h () [constexpr]
Literal suffix for durations of type std::chrono::hours
```

**operator""h() [2/2]**

```
chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (
    long double __hours) [constexpr]
Literal suffix for durations representing non-integer hours.
```

**operator""min() [1/2]**

```
template<char... _Digits>
chrono::minutes std::literals::chrono_literals::operator""min () [constexpr]
Literal suffix for durations of type std::chrono::minutes
```

**operator""min() [2/2]**

```
chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (
    long double __mins) [constexpr]
Literal suffix for durations representing non-integer minutes.
```

**operator""ms() [1/2]**

```
template<char... _Digits>
chrono::milliseconds std::literals::chrono_literals::operator""ms () [constexpr]
Literal suffix for durations of type std::chrono::milliseconds
```

**operator""ms() [2/2]**

```
chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (
    long double __msecs) [constexpr]
Literal suffix for durations representing non-integer milliseconds.
```

**operator""ns() [1/2]**

```
template<char... _Digits>
chrono::nanoseconds std::literals::chrono_literals::operator""ns () [constexpr]
Literal suffix for durations of type std::chrono::nanoseconds
```

**operator""ns() [2/2]**

```
chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (
    long double __nsecs) [constexpr]
Literal suffix for durations representing non-integer nanoseconds.
```

**operator""s() [1/2]**

```
template<char... _Digits>
chrono::seconds std::literals::chrono_literals::operator""s () [constexpr]
Literal suffix for durations of type std::chrono::seconds
```

**operator""s()** [2/2]

```
chrono::duration< long double > std::literals::chrono_literals::operator""s (
    long double __secs) [constexpr]
```

Literal suffix for durations representing non-integer seconds.

**operator""us()** [1/2]

```
template<char... _Digits>
chrono::microseconds std::literals::chrono_literals::operator""us () [constexpr]
Literal suffix for durations of type std::chrono::microseconds
```

**operator""us()** [2/2]

```
chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (
    long double __usecs) [constexpr]
```

Literal suffix for durations representing non-integer microseconds.

**operator""y()**

```
chrono::year std::literals::chrono_literals::operator""y (
    unsigned long long __y) [constexpr], [noexcept]
```

Literal suffix for creating chrono::year objects.

Since

C++20

**operator%()** [1/2]

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono↵
::operator% (
    const duration< _Rep1, _Period > & __d,
    const _Rep2 & __s) [constexpr]
```

Arithmetic operators for chrono::duration

**operator%()** [2/2]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono↵
::operator% (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Arithmetic operators for chrono::duration

**operator\*()** [1/2]

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (
    const duration< _Rep1, _Period > & __d,
    const _Rep2 & __s) [related]
```

Arithmetic operators for chrono::duration

**operator\*() [2/2]**

```
template<typename _Rep1, typename _Rep2, typename _Period>
duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (
    const _Rep1 & __s,
    const duration< _Rep2, _Period > & __d) [constexpr]
```

Arithmetic operators for chrono::duration

**operator+() [1/3]**

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [related]
```

The sum of two durations.

**operator+() [2/3]**

```
template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > std::
::chrono::operator+ (
    const duration< _Rep1, _Period1 > & __lhs,
    const time_point< _Clock, _Dur2 > & __rhs) [constexpr]
```

Adjust a time point forwards by the given duration.

**operator+() [3/3]**

```
template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+
(
    const time_point< _Clock, _Dur1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [related]
```

Adjust a time point forwards by the given duration.

**operator-() [1/3]**

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::
operator- (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

The difference between two durations.

**operator-() [2/3]**

```
template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > std::
::chrono::operator- (
    const time_point< _Clock, _Dur1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Adjust a time point backwards by the given duration.

**operator-() [3/3]**

```
template<typename _Clock, typename _Dur1, typename _Dur2>
common_type< _Dur1, _Dur2 >::type std::chrono::operator- (
```

```
const time_point< _Clock, _Dur1 > & __lhs,
const time_point< _Clock, _Dur2 > & __rhs) [constexpr]
```

The difference between two time points (as a duration)

#### operator/() [1/2]

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator/ (
    const duration< _Rep1, _Period > & __d,
    const _Rep2 & __s) [constexpr]
```

Arithmetic operators for chrono::duration

#### operator/() [2/2]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Arithmetic operators for chrono::duration

#### operator<()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator< (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for chrono::duration

#### operator<<() [1/3]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const local_info & __li)
```

Writes a local\_info object to an ostream in an unspecified format.

#### operator<<() [2/3]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const sys_info & __i)
```

Writes a sys\_info object to an ostream in an unspecified format.

#### operator<<() [3/3]

```
template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const duration< _Rep, _Period > & __d) [inline]
```

Write a chrono::duration to an ostream.



Since

C++20

References [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::imbue\(\)](#), [std::move\(\)](#), [operator<<\(\)](#), and [std::ios\\_base::precision\(\)](#).

Referenced by [operator<<\(\)](#).

### **operator<=()**

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator<= (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for `chrono::duration`

### **operator<=>()**

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
requires three_way_comparable<common_type_t<_Rep1, _Rep2>>
auto std::chrono::operator<=> (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for `chrono::duration`

### **operator==()**

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool operator== (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [related]
```

Comparisons for `chrono::duration`

### **operator>()**

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator> (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for `chrono::duration`

### **operator>=()**

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator>= (
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for `chrono::duration`

### **round()** [1/2]

```
template<typename _ToDur, typename _Rep, typename _Period>
enable_if_t< __and< __is_duration< _ToDur >, __not< treat_as_floating_point< typename _ToDur<
::rep > > >::value, _ToDur > std::chrono::round (
    const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type `ToDur` and round to the closest value.

If the duration cannot be represented exactly in the result type, returns the closest value, rounding ties to even.

## Template Parameters

<code>_ToDur</code>	The result type must be a <code>duration</code> with a non-floating-point <code>rep</code> type.
---------------------	--

## Parameters

<code>__d</code>	A duration.
------------------	-------------

## Returns

The value of `__d` converted to type `_ToDur`.

## Since

C++17

References [floor\(\)](#).

Referenced by [round\(\)](#).

**round()** [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >,
time_point< _Clock, _ToDur > > std::chrono::round (
    const time_point< _Clock, _Dur > & __tp) [nodiscard], [constexpr]
```

Convert a `time_point` to type `ToDur` and round to the closest value.

The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value, rounding ties to even.

## Template Parameters

<code>_ToDur</code>	The <code>duration</code> type to use for the result, which must have a non-floating-point <code>rep</code> type.
---------------------	---

## Parameters

<code>t</code>	A time point.
----------------	---------------

## Returns

The value of `__d` converted to type `_ToDur`.

## Since

C++17

References [round\(\)](#).

**time\_point\_cast()**

```
template<typename _ToDur, typename _Clock, typename _Dur>
__enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono<
::time_point_cast (
    const time_point< _Clock, _Dur > & __t) [nodiscard], [constexpr]
```

Convert a `time_point` to use duration type `ToDur`.

The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented accurately in the result type, returns the result of integer truncation (i.e., rounded towards zero).

**Template Parameters**

<code>_ToDur</code>	The <code>duration</code> type to use for the result.
---------------------	---

**Parameters**

<code>__t</code>	A time point.
------------------	---------------

**Returns**

The value of `__t` converted to use type `_ToDur`.

**Since**

C++11

References [duration\\_cast\(\)](#).

## 4 Namespace Documentation

### 4.1 `__gnu_cxx` Namespace Reference

**Namespaces**

- namespace [\\_\\_detail](#)
- namespace [typelist](#)

**Classes**

- struct [\\_\\_alloc\\_traits](#)
- struct [\\_\\_common\\_pool\\_policy](#)
- class [\\_\\_mt\\_alloc](#)
- class [\\_\\_mt\\_alloc\\_base](#)
- struct [\\_\\_per\\_type\\_pool\\_policy](#)
- class [\\_\\_pool](#)
- class [\\_\\_pool< false >](#)
- class [\\_\\_pool< true >](#)
- class [\\_\\_pool\\_alloc](#)

- class [\\_\\_pool\\_alloc\\_base](#)
- struct [\\_\\_pool\\_base](#)
- class [\\_\\_rc\\_string\\_base](#)
- class [\\_\\_scoped\\_lock](#)
- class [\\_\\_versa\\_string](#)
- struct [\\_Caster](#)
- struct [\\_Char\\_types](#)
- class [\\_ExtPtr\\_allocator](#)
- struct [\\_Invalid\\_type](#)
- class [\\_Pointer\\_adapter](#)
- class [\\_Relative\\_pointer\\_impl](#)
- class [\\_Relative\\_pointer\\_impl< const \\_Tp >](#)
- class [\\_Std\\_pointer\\_impl](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Unqualified\\_type](#)
- class [allocator](#)
- struct [annotate\\_base](#)
- class [binary\\_compose](#)
- class [bitmap\\_allocator](#)
- struct [char\\_traits](#)
- struct [character](#)
- struct [condition\\_base](#)
- struct [constant\\_binary\\_fun](#)
- struct [constant\\_unary\\_fun](#)
- struct [constant\\_void\\_fun](#)
- class [debug\\_allocator](#)
- class [enc\\_filebuf](#)
- struct [encoding\\_char\\_traits](#)
- class [encoding\\_state](#)
- struct [equal\\_to](#)
- struct [forced\\_error](#)
- class [free\\_list](#)
- class [hash\\_map](#)
- class [hash\\_multimap](#)
- class [hash\\_multiset](#)
- class [hash\\_set](#)
- struct [limit\\_condition](#)
- class [malloc\\_allocator](#)
- class [new\\_allocator](#)
- struct [pair](#)
- struct [project1st](#)
- struct [project2nd](#)
- struct [random\\_condition](#)
- struct [rb\\_tree](#)
- class [recursive\\_init\\_error](#)
- class [rope](#)
- struct [select1st](#)
- struct [select2nd](#)
- class [slist](#)
- class [stdio\\_filebuf](#)
- class [stdio\\_sync\\_filebuf](#)

- class `subtractive_rng`
- struct `temporary_buffer`
- class `throw_allocator_base`
- struct `throw_allocator_limit`
- struct `throw_allocator_random`
- struct `throw_value_base`
- struct `throw_value_limit`
- struct `throw_value_random`
- class `unary_compose`

## Typedefs

- typedef void(\* `__destroy_handler`) (void \*)
- template<typename `_Tp`>  
using `__int_traits`
- typedef `__versa_string`< char, `std::char_traits`< char >, `std::allocator`< char >, `__rc_string_base` > `__rc_string`
- typedef `__vstring` `__sso_string`
- typedef `__versa_string`< char16\_t, `std::char_traits`< char16\_t >, `std::allocator`< char16\_t >, `__rc_string_base` > `__u16rc_string`
- typedef `__u16vstring` `__u16sso_string`
- typedef `__versa_string`< char16\_t > `__u16vstring`
- typedef `__versa_string`< char32\_t, `std::char_traits`< char32\_t >, `std::allocator`< char32\_t >, `__rc_string_base` > `__u32rc_string`
- typedef `__u32vstring` `__u32sso_string`
- typedef `__versa_string`< char32\_t > `__u32vstring`
- typedef `__versa_string`< char > `__vstring`
- typedef `__versa_string`< wchar\_t, `std::char_traits`< wchar\_t >, `std::allocator`< wchar\_t >, `__rc_string_base` > `__wrc_string`
- typedef `__wvstring` `__wsso_string`
- typedef `__versa_string`< wchar\_t > `__wvstring`
- typedef `rope`< char > `crope`
- typedef `__SIZE_TYPE__` `__size_t`
- typedef `rope`< wchar\_t > `wrope`

## Enumerations

- enum { `_S_num_primes` }
- enum `_Lock_policy` { `_S_single` , `_S_mutex` , `_S_atomic` }

## Functions

- void `__atomic_add` (volatile `_Atomic_word` \* `__mem`, int `__val`)
- void `__atomic_add_dispatch` (`_Atomic_word` \* `__mem`, int `__val`)
- void `__atomic_add_single` (`_Atomic_word` \* `__mem`, int `__val`)
- template<class `_Tp`>  
void `__aux_require_boolean_expr` (const `_Tp` & `__t`)
- template<typename `_ToType`, typename `_FromType`>  
`_ToType` `__const_pointer_cast` (`_FromType` \* `__arg`)
- template<typename `_ToType`, typename `_FromType`>  
`_ToType` `__const_pointer_cast` (const `_FromType` & `__arg`)
- template<typename `_InputIterator`, typename `_Size`, typename `_OutputIterator`>  
`std::pair`< `_InputIterator`, `_OutputIterator` > `__copy_n` (`_InputIterator` `__first`, `_Size` `__count`, `_OutputIterator` `__result`, `std::input_iterator_tag`)

- `template<typename _RAIterator, typename _Size, typename _OutputIterator>`  
`std::pair<_RAIterator, _OutputIterator> __copy_n(_RAIterator __first, _Size __count, _OutputIterator __result, std::random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Distance>`  
`void __distance(_InputIterator __first, _InputIterator __last, _Distance &__n, std::input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance>`  
`void __distance(_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random_access_iterator_tag)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __dynamic_pointer_cast(_FromType * __arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __dynamic_pointer_cast(const _FromType &__arg)`
- `void __error_type_must_be_a_signed_integer_type()`
- `void __error_type_must_be_an_integer_type()`
- `void __error_type_must_be_an_unsigned_integer_type()`
- `_Atomic_word __exchange_and_add(volatile _Atomic_word * __mem, int __val)`
- `_Atomic_word __exchange_and_add_dispatch(_Atomic_word * __mem, int __val)`
- `_Atomic_word __exchange_and_add_single(_Atomic_word * __mem, int __val)`
- `template<class _Concept>`  
`constexpr void __function_requires()`
- `template<typename _Type>`  
`constexpr bool __is_null_pointer(_Type * __ptr)`
- `template<typename _Type>`  
`constexpr bool __is_null_pointer(_Type)`
- `constexpr bool __is_null_pointer(std::nullptr_t)`
- `bool __is_single_threaded() noexcept`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`int __lexicographical_compare_3way(_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `int __lexicographical_compare_3way(const char * __first1, const char * __last1, const char * __first2, const char * __last2)`
- `int __lexicographical_compare_3way(const unsigned char * __first1, const unsigned char * __last1, const unsigned char * __first2, const unsigned char * __last2)`
- `template<typename _Tp>`  
`const _Tp & __median(const _Tp & __a, const _Tp & __b, const _Tp & __c)`
- `template<typename _Tp, typename _Compare>`  
`const _Tp & __median(const _Tp & __a, const _Tp & __b, const _Tp & __c, _Compare __comp)`
- `crope::reference __mutable_reference_at(crope & __c, std::size_t __i)`
- `template<typename _Tp, typename _Integer>`  
`_Tp __power(_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`  
`_Tp __power(_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance>`  
`_RandomAccessIterator __random_sample(_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator & __rand, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance>`  
`_RandomAccessIterator __random_sample(_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __reinterpret_pointer_cast(_FromType * __arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __reinterpret_pointer_cast(const _FromType & __arg)`
- `_Slist_node_base * __slist_make_link(_Slist_node_base * __prev_node, _Slist_node_base * __new_node)`

- `_Slist_node_base * __slist_previous` (`_Slist_node_base * __head`, `const _Slist_node_base * __node`)
- `const _Slist_node_base * __slist_previous` (`const _Slist_node_base * __head`, `const _Slist_node_base * __node`)
- `_Slist_node_base * __slist_reverse` (`_Slist_node_base * __node`)
- `std::size_t __slist_size` (`_Slist_node_base * __node`)
- `void __slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __before_first`, `_Slist_node_base * __before_last`)
- `void __slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __head`)
- `template<typename _ToType, typename _FromType>`  
`_ToType __static_pointer_cast` (`_FromType * __arg`)
- `template<typename _ToType, typename _FromType>`  
`_ToType __static_pointer_cast` (`const _FromType & __arg`)
- `size_t __stl_hash_string` (`const char * __s`)
- `unsigned long __stl_next_prime` (`unsigned long __n`)
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`  
`_Ret __stoa` (`_TRet(* __convf)`(`const _CharT *`, `_CharT **`, `_Base...`), `const char * __name`, `const _CharT * __str`, `std::size_t * __idx`, `_Base... __base`)
- `void __throw_concurrency_lock_error` ()
- `void __throw_concurrency_unlock_error` ()
- `void __throw_forced_error` ()
- `template<typename _String, typename _CharT = typename _String::value_type>`  
`_String __to_xstring` (`int(* __convf)`(`_CharT *`, `std::size_t`, `const _CharT *`, `__builtin_va_list`), `std::size_t __n`, `const _CharT * __fmt`,...)
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair< _InputIter, _ForwardIter > __uninitialized_copy_n` (`_InputIter __first`, `_Size __count`, `_ForwardIter __result`)
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair< _InputIter, _ForwardIter > __uninitialized_copy_n` (`_InputIter __first`, `_Size __count`, `_ForwardIter __result`, `std::input_iterator_tag`)
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter>`  
`std::pair< _RandomAccessIter, _ForwardIter > __uninitialized_copy_n` (`_RandomAccessIter __first`, `_Size __count`, `_ForwardIter __result`, `std::random_access_iterator_tag`)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator>`  
`std::pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a` (`_InputIter __first`, `_Size __count`, `_ForwardIter __result`, `_Allocator __alloc`)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp>`  
`std::pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a` (`_InputIter __first`, `_Size __count`, `_ForwardIter __result`, `std::allocator< _Tp >`)
- `void __verbose_terminate_handler` ()
- `std::size_t __Bit_scan_forward` (`std::size_t __num`)
- `template<typename _ForwardIterator, typename _Allocator>`  
`void __Destroy_const` (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Allocator __alloc`)
- `template<typename _ForwardIterator, typename _Tp>`  
`void __Destroy_const` (`_ForwardIterator __first`, `_ForwardIterator __last`, `std::allocator< _Tp >`)
- `template<class _CharT, class _Traits>`  
`void __Rope_fill` (`std::basic_ostream< _CharT, _Traits > & __o`, `std::size_t __n`)
- `template<class _CharT>`  
`bool __Rope_is_simple` (`_CharT *`)
- `bool __Rope_is_simple` (`char *`)
- `bool __Rope_is_simple` (`wchar_t *`)
- `template<class _Rope_iterator>`  
`void __Rope_rotate` (`_Rope_iterator __first`, `_Rope_iterator __middle`, `_Rope_iterator __last`)



- `template<class _CharT>`  
`void _S_cond_store_eos (_CharT &)`
- `void _S_cond_store_eos (char &__c)`
- `void _S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT>`  
`_CharT _S_eos (_CharT *)`
- `template<class _CharT>`  
`bool _S_is_basic_char_type (_CharT *)`
- `bool _S_is_basic_char_type (char *)`
- `bool _S_is_basic_char_type (wchar_t *)`
- `template<class _CharT>`  
`bool _S_is_one_byte_char_type (_CharT *)`
- `bool _S_is_one_byte_char_type (char *)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type airy_ai (_Tp __x)`
- `float airy_aif (float __x)`
- `long double airy_ail (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type airy_bi (_Tp __x)`
- `float airy_bif (float __x)`
- `long double airy_bil (long double __x)`
- `template<class _Operation1, class _Operation2>`  
`unary_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3>`  
`binary_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<typename _Tpa, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float conf_hypergf (float __a, float __c, float __x)`
- `long double conf_hypergl (long double __a, long double __c, long double __x)`
- `template<class _Result>`  
`constant_void_fun< _Result > constant0 (const _Result &__val)`
- `template<class _Result>`  
`constant_unary_fun< _Result, _Result > constant1 (const _Result &__val)`
- `template<class _Result>`  
`constant_binary_fun< _Result, _Result, _Result > constant2 (const _Result &__val)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`std::pair< _InputIterator, _OutputIterator > copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size>`  
`void count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size>`  
`void count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator, typename _Distance>`  
`void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float hypergf (float __a, float __b, float __c, float __x)`
- `long double hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<class _Tp>`  
`_Tp identity_element (std::multiplies< _Tp >)`

- `template<class _Tp>`  
`_Tp identity_element (std::plus< _Tp >)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RAIter>`  
`constexpr bool is_heap (_RAIter, _RAIter)`
- `template<typename _Filter>`  
`constexpr bool is_sorted (_Filter, _Filter)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`int lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<class _Ret, class _Tp, class _Arg>`  
`std::const_mem_fun1_t< _Ret, _Tp, _Arg > mem_fun1 (_Ret(_Tp::*__f)( _Arg) const)`
- `template<class _Ret, class _Tp, class _Arg>`  
`std::mem_fun1_t< _Ret, _Tp, _Arg > mem_fun1 (_Ret(_Tp::*__f)( _Arg))`
- `template<class _Ret, class _Tp, class _Arg>`  
`std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun1_ref (_Ret(_Tp::*__f)( _Arg) const)`
- `template<class _Ret, class _Tp, class _Arg>`  
`std::mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun1_ref (_Ret(_Tp::*__f)( _Arg))`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp>`  
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp>`  
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator!= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>`  
`bool operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>`  
`bool operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`

- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>`  
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _CharT, class _Alloc>`  
`bool operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _Tp, class _Alloc>`  
`bool operator!= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _Tp>`  
`bool operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Cond>`  
`throw_value_base< _Cond > operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base > && __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > && __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base > && __lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base > && __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base > && __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( _CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > && __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( _CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > && __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > && __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT * __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > && __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > && __rhs)`
- `template<class _CharT, class _Alloc>`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`

- `template<class _CharT, class _Alloc>`  
`_Rope_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<typename _Cond>`  
`throw_value_base< _Cond > operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _CharT, class _Alloc>`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (std::ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc>`  
`_Rope_iterator< _CharT, _Alloc > operator+ (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc>`  
`std::ptrdiff_t operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`_Rope_const_iterator< _CharT, _Alloc > operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>`  
`std::ptrdiff_t operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`_Rope_iterator< _CharT, _Alloc > operator- (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<typename _Cond>`  
`throw_value_base< _Cond > operator- (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc>`  
`bool operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Value, typename _Int, typename _St>`  
`bool operator< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _Tp, class _Alloc>`  
`bool operator< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _Cond>`  
`bool operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _CharT, class _Traits, class _Alloc>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<typename _UIntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::triangular_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::von_mises_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _StoreT>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const k_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`logistic_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`nakagami_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`rice_distribution< _RealType > &__x)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _↵`  
`_CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp>`  
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT,`  
`_Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &↵`  
`__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _Tp, class _Alloc>`  
`bool operator<= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<size_t _Dimen, typename _RealType>`  
`bool operator== (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const __gnu_cxx↵`  
`::normal_mv_distribution< _Dimen, _RealType > &__d2)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`  
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>`  
`bool operator== (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __↵`  
`sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__lhs, const`  
`__gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1,`  
`__msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)`
- `template<typename _Tp, typename _Poolp>`  
`bool operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp>`  
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT,`  
`_Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator== (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp>`  
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp>`  
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT,`  
`_Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT,`  
`_Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _Value, typename _Int, typename _St>`  
`bool operator== (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>`  
`bool operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key,`  
`_Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>`  
`bool operator== (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap<`  
`_Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset<`  
`_Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value,`  
`_HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>`  
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key,`  
`_HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _CharT, class _Alloc>`  
`bool operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _Tp, class _Alloc>`  
`bool operator== (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _Tp, typename _Cond>`  
`bool operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Cond>`  
`bool operator== (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Tp>`  
`bool operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`



- `template<typename _Tp1, typename _Tp2>`  
`bool operator> (_Tp1 __lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp>`  
`bool operator> (const \_Pointer\_adapter< _Tp > &__lhs, const \_Pointer\_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator> (const \_Pointer\_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator> (const \_Pointer\_adapter< _Tp1 > &__lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _Tp, class _Alloc>`  
`bool operator> (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator>= (_Tp1 __lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp>`  
`bool operator>= (const \_Pointer\_adapter< _Tp > &__lhs, const \_Pointer\_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator>= (const \_Pointer\_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator>= (const \_Pointer\_adapter< _Tp1 > &__lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _Tp, class _Alloc>`  
`bool operator>= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > &operator>> (std::basic\_istream< _CharT, _Traits > &__is, __gnu_cxx::beta_distribution< _RealType > &__x)`



- `template<typename _UIntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::triangular_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::von_mises_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, k_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, logistic_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, nakagami_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, rice_distribution< _RealType > &__x)`
- `template<typename _Tp, typename _Integer>`  
`_Tp power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`  
`_Tp power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`

- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`_RandomAccessIterator random_sample ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator <←`  
`__out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>`  
`_OutputIterator random_sample_n ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const`  
`_Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>`  
`_OutputIterator random_sample_n ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const`  
`_Distance __n, _RandomNumberGenerator &__rand)`
- `void rotate ( _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __`  
`STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)>`  
`__last)`
- `double stod (const __vstring &__str, std::size_t * __idx=0)`
- `double stod (const __wvstring &__str, std::size_t * __idx=0)`
- `float stof (const __vstring &__str, std::size_t * __idx=0)`
- `float stof (const __wvstring &__str, std::size_t * __idx=0)`
- `int stoi (const __vstring &__str, std::size_t * __idx=0, int __base=10)`
- `int stoi (const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `long stol (const __vstring &__str, std::size_t * __idx=0, int __base=10)`
- `long stol (const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `long double stold (const __vstring &__str, std::size_t * __idx=0)`
- `long double stold (const __wvstring &__str, std::size_t * __idx=0)`
- `long long stoll (const __vstring &__str, std::size_t * __idx=0, int __base=10)`
- `long long stoll (const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `unsigned long stoul (const __vstring &__str, std::size_t * __idx=0, int __base=10)`
- `unsigned long stoul (const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `unsigned long long stoull (const __vstring &__str, std::size_t * __idx, int __base=10)`
- `unsigned long long stoull (const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`void swap ( __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc,`  
`_Base > &__rhs)`
- `template<typename _Tp>`  
`void swap ( _ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)`
- `template<class _CharT, class _Alloc>`  
`void swap ( _Rope_char_ref_proxy< _CharT, _Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>`  
`void swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn,`  
`_EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>`  
`void swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp,`  
`_HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`void swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn,`  
`_EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`void swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey,`  
`_Alloc > &__hs2)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All>`  
`void swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract,`  
`_EqKey, _All > &__ht2)`
- `template<class _CharT, class _Alloc>`  
`void swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`

- `template<class _Tp, class _Alloc>`  
`void swap(slist<_Tp, _Alloc> &__x, slist<_Tp, _Alloc> &__y)`
- `template<typename _Cond>`  
`void swap(throw_value_base<_Cond> &__a, throw_value_base<_Cond> &__b)`
- `__vstring to_string` (double \_\_val)
- `__vstring to_string` (float \_\_val)
- `__vstring to_string` (int \_\_val)
- `__vstring to_string` (long \_\_val)
- `__vstring to_string` (long double \_\_val)
- `__vstring to_string` (long long \_\_val)
- `__vstring to_string` (unsigned \_\_val)
- `__vstring to_string` (unsigned long \_\_val)
- `__vstring to_string` (unsigned long long \_\_val)
- `__wvstring to_wstring` (double \_\_val)
- `__wvstring to_wstring` (float \_\_val)
- `__wvstring to_wstring` (int \_\_val)
- `__wvstring to_wstring` (long \_\_val)
- `__wvstring to_wstring` (long double \_\_val)
- `__wvstring to_wstring` (long long \_\_val)
- `__wvstring to_wstring` (unsigned \_\_val)
- `__wvstring to_wstring` (unsigned long \_\_val)
- `__wvstring to_wstring` (unsigned long long \_\_val)
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair<_InputIter, _ForwardIter> uninitialized_copy_n(_InputIter __first, _Size __count, _ForwardIter __↵  
result)`

## Variables

- `const _Lock_policy __default_lock_policy`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__gamma_e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__ln_10`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__ln_2`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__ln_3`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__log10_e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__log2_e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__one_div_e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__one_div_pi`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__one_div_root_2`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants<_RealType>::__phi`

- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__pi`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__pi_half`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__pi_quarter`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__pi_third`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_2`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_3`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_5`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_7`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_pi_div_2`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__two_div_pi`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__two_div_root_pi`
- `template<typename _Value>`  
`const int __numeric_traits_floating< _Value >::__digits10`
- `template<typename _Value>`  
`const bool __numeric_traits_floating< _Value >::__is_signed`
- `template<typename _Value>`  
`const int __numeric_traits_floating< _Value >::__max_digits10`
- `template<typename _Value>`  
`const int __numeric_traits_floating< _Value >::__max_exponent10`
- `template<typename _Value>`  
`const int __numeric_traits_integer< _Value >::__digits`
- `template<typename _Value>`  
`const bool __numeric_traits_integer< _Value >::__is_signed`
- `template<typename _Value>`  
`const _Value __numeric_traits_integer< _Value >::__max`
- `template<typename _Value>`  
`const _Value __numeric_traits_integer< _Value >::__min`
- `template<typename _Tp>`  
`_Atomic_word __pool_alloc< _Tp >::__S_force_new`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`__rc_string_base< _CharT, _Traits, _Alloc >::__Rep_empty __rc_string_base< _CharT, _Traits, _Alloc >::__S_empty_rep`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __versa_string< _CharT, _Traits, _Alloc, _Base >::npos`
- `template<typename _PrimeType>`  
`const _PrimeType __Hashtable_prime_list< _PrimeType >::__stl_prime_list [ _S_num_primes]`
- `template<typename _Tp>`  
`std::size_t bitmap_allocator< _Tp >::__S_block_size`
- `template<typename _Tp>`  
`bitmap_allocator< _Tp >::__BPVector::size_type bitmap_allocator< _Tp >::__S_last_dealloc_index`

- `template<typename _Tp>`  
`bitmap_allocator< _Tp >::_BPVector bitmap_allocator< _Tp >::_S_mem_blocks`
- `template<typename _Tp>`  
`bitmap_allocator< _Tp >::__mutex_type bitmap_allocator< _Tp >::_S_mut`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`
- `template<class _CharT, class _Alloc>`  
`_CharT rope< _CharT, _Alloc >::_S_empty_c_str [1]`
- `template<class _CharT, class _Alloc>`  
`const unsigned long rope< _CharT, _Alloc >::_S_min_len [int(__detail::_S_max_rope_depth)+1]`
- `template<class _CharT, class _Alloc>`  
`const rope< _CharT, _Alloc >::size_type rope< _CharT, _Alloc >::npos`

#### 4.1.1 Detailed Description

GNU extensions for public use.

#### 4.1.2 Typedef Documentation

##### `__int_traits`

```
template<typename _Tp>
using __gnu_cxx::__int_traits
Convenience alias for __numeric_traits<integer-type>.
```

#### 4.1.3 Function Documentation

##### `__static_pointer_cast()` [1/2]

```
template<typename _ToType, typename _FromType>
_ToType __gnu_cxx::__static_pointer_cast (
    _FromType * __arg) [inline]
```

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

##### `__static_pointer_cast()` [2/2]

```
template<typename _ToType, typename _FromType>
_ToType __gnu_cxx::__static_pointer_cast (
    const _FromType & __arg) [inline]
```

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

##### `_Bit_scan_forward()`

```
std::size_t __gnu_cxx::_Bit_scan_forward (
    std::size_t __num) [inline]
```

Generic Version of the bsf instruction.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp1 >::_M_allocate_single_object()`.

##### `iota()`

```
template<typename _ForwardIterator, typename _Tp>
void std::iota (
    _ForwardIterator __first,
```

```

    _ForwardIterator __last,
    _Tp __value) [constexpr]

```

Create a range of sequentially increasing values.

For each element in the range `[first,last)` assigns `value` and increments `value` as if by `++value`.

#### Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__value</code>	Starting value.

#### Returns

Nothing.

#### `operator"!=()` [1/3]

```

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]

```

Test difference of two strings.

#### Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

#### Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

#### `operator"!=()` [2/3]

```

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs) [inline]

```

Test difference of string and C string.

#### Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

#### Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

**operator!=(())** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!=(
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test difference of C string and string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__rhs.compare(__lhs) != 0`. False otherwise.

**operator+()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    _CharT __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate character and string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::append\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::reserve\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::operator+\(\)](#).

**operator+()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    _CharT __rhs)
```

Concatenate string and character.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::append\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::reserve\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::size\(\)](#).

**operator+()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate two strings.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with value of `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::append\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::size\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::size\(\)](#).

**operator+()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs)
```

Concatenate string and C string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::size\(\)](#).

**operator+()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate C string and string.



**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with value of `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::size\(\)](#).

**operator<()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string precedes string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator<()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs) [inline]
```

Test if string precedes C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator<()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string precedes string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator<=()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string doesn't follow string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator<=()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs) [inline]
```

Test if string doesn't follow C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::compare\(\)](#).

**operator<=()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string doesn't follow string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::compare\(\)](#).

**operator==( )** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator==(
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test equivalence of two strings.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::data\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_ALL](#)

**operator==( )** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator==(
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs) [inline]
```

Test equivalence of string and C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::data\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::compare\(\)](#).

**operator==()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator== (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test equivalence of C string and string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

**operator==()** [4/4]

```
template<typename _Tp>
bool __gnu_cxx::operator== (
    const _Pointer_adapter< _Tp > & __lhs,
    const _Pointer_adapter< _Tp > & __rhs) [inline]
```

Comparison operators for `_Pointer_adapter` defer to the base class' comparison operators, when possible.

**operator>()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string follows string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::compare\(\)](#).

**operator>()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs) [inline]
```

Test if string follows C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator>()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string follows string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator>=()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string doesn't precede string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::compare\(\)](#).

**operator>=()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs) [inline]
```

Test if string doesn't precede C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::compare\(\)](#).

**operator>=()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string doesn't precede string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::compare\(\)](#).

**swap()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::swap (
    __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Swap contents of two strings.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::swap\(\)](#).

## 4.2 \_\_gnu\_cxx::\_\_detail Namespace Reference

### Classes

- class [\\_\\_mini\\_vector](#)
- class [\\_Bitmap\\_counter](#)
- class [\\_Ffit\\_finder](#)

### Enumerations

- enum { `_S_max_rope_depth` }
- enum { `bits_per_byte` , `bits_per_block` }
- enum `_Tag` { `_S_leaf` , `_S_concat` , `_S_substringfn` , `_S_function` }

### Functions

- void [\\_\\_bit\\_allocate](#) (std::size\_t \* \_\_pmap, std::size\_t \_\_pos) throw ()
- void [\\_\\_bit\\_free](#) (std::size\_t \* \_\_pmap, std::size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare>  
\_ForwardIterator [\\_\\_lower\\_bound](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair>  
std::size\_t [\\_\\_num\\_bitmaps](#) (\_AddrPair \_\_ap)
- template<typename \_AddrPair>  
std::size\_t [\\_\\_num\\_blocks](#) (\_AddrPair \_\_ap)

#### 4.2.1 Detailed Description

Implementation details not part of the namespace `__gnu_cxx` interface.

#### 4.2.2 Function Documentation

##### [\\_\\_bit\\_allocate\(\)](#)

```
void __gnu_cxx::__detail::__bit_allocate (
    std::size_t * __pmap,
    std::size_t __pos) throw ( )    [inline]
```

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Referenced by [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp1 >::\\_M\\_allocate\\_single\\_object\(\)](#).

##### [\\_\\_bit\\_free\(\)](#)

```
void __gnu_cxx::__detail::__bit_free (
    std::size_t * __pmap,
    std::size_t __pos) throw ( )    [inline]
```

Mark a memory address as free by setting the corresponding bit in the bit-map.

Referenced by [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp1 >::\\_M\\_deallocate\\_single\\_object\(\)](#).

**`__num_bitmaps()`**

```
template<typename _AddrPair>
std::size_t __gnu_cxx::__detail::__num_bitmaps (
    _AddrPair __ap) [inline]
```

The number of Bit-maps pointed to by the address pair passed to the function.

References [\\_\\_num\\_blocks\(\)](#).

Referenced by [\\_\\_gnu\\_cxx::bitmap\\_allocator< \\_Tp1 >::\\_M\\_allocate\\_single\\_object\(\)](#), and [\\_\\_gnu\\_cxx::bitmap\\_allocator< \\_Tp1 >::\\_M\\_deallocate\(\)](#).

**`__num_blocks()`**

```
template<typename _AddrPair>
std::size_t __gnu_cxx::__detail::__num_blocks (
    _AddrPair __ap) [inline]
```

The number of Blocks pointed to by the address pair passed to the function.

Referenced by [\\_\\_num\\_bitmaps\(\)](#).

**4.3 `__gnu_cxx::typelist` Namespace Reference****Functions**

- `template<typename Fn, typename Typelist>`  
`void apply (Fn &, Typelist)`
- `template<typename Fn, typename Typelist>`  
`void apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV>`  
`void apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Gn, typename Typelist>`  
`void apply\_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV>`  
`void apply_generator (Gn &, TypelistT, TypelistV)`

**4.3.1 Detailed Description**

GNU typelist extensions for public compile-time use.

**4.3.2 Function Documentation****`apply_generator()`**

```
template<typename Gn, typename Typelist>
void __gnu_cxx::typelist::apply_generator (
    Gn &,
    Typelist )
```

Apply all typelist types to generator functor.

**4.4 `__gnu_debug` Namespace Reference****Classes**

- class [\\_\\_After\\_nth\\_from](#)
- struct [\\_\\_BeforeBeginHelper](#)
- class [\\_\\_Equal\\_to](#)
- class [\\_\\_Not\\_equal\\_to](#)
- class [\\_\\_Safe\\_container](#)
- class [\\_\\_Safe\\_forward\\_list](#)



- class [\\_Safe\\_iterator](#)
- class [\\_Safe\\_iterator\\_base](#)
- class [\\_Safe\\_local\\_iterator](#)
- class [\\_Safe\\_local\\_iterator\\_base](#)
- class [\\_Safe\\_node\\_sequence](#)
- class [\\_Safe\\_sequence](#)
- class [\\_Safe\\_sequence\\_base](#)
- class [\\_Safe\\_unordered\\_container](#)
- class [\\_Safe\\_unordered\\_container\\_base](#)
- class [\\_Safe\\_vector](#)
- struct [\\_Sequence\\_traits](#)
- class [basic\\_string](#)
- class [type\\_info](#)

## Typedefs

- typedef [basic\\_string](#)< char > **string**
- typedef [basic\\_string](#)< char16\_t > **u16string**
- typedef [basic\\_string](#)< char32\_t > **u32string**
- typedef [basic\\_string](#)< wchar\_t > **wstring**

## Enumerations

- enum [\\_Debug\\_msg\\_id](#) {  
[\\_\\_msg\\_valid\\_range](#) , [\\_\\_msg\\_insert\\_singular](#) , [\\_\\_msg\\_insert\\_different](#) , [\\_\\_msg\\_erase\\_bad](#) ,  
[\\_\\_msg\\_erase\\_different](#) , [\\_\\_msg\\_subscript\\_oob](#) , [\\_\\_msg\\_empty](#) , [\\_\\_msg\\_unpartitioned](#) ,  
[\\_\\_msg\\_unpartitioned\\_pred](#) , [\\_\\_msg\\_unsorted](#) , [\\_\\_msg\\_unsorted\\_pred](#) , [\\_\\_msg\\_not\\_heap](#) ,  
[\\_\\_msg\\_not\\_heap\\_pred](#) , [\\_\\_msg\\_bad\\_bitset\\_write](#) , [\\_\\_msg\\_bad\\_bitset\\_read](#) , [\\_\\_msg\\_bad\\_bitset\\_flip](#) ,  
[\\_\\_msg\\_self\\_splice](#) , [\\_\\_msg\\_splice\\_alloc](#) , [\\_\\_msg\\_splice\\_bad](#) , [\\_\\_msg\\_splice\\_other](#) ,  
[\\_\\_msg\\_splice\\_overlap](#) , [\\_\\_msg\\_init\\_singular](#) , [\\_\\_msg\\_init\\_copy\\_singular](#) , [\\_\\_msg\\_init\\_const\\_singular](#) ,  
[\\_\\_msg\\_copy\\_singular](#) , [\\_\\_msg\\_bad\\_deref](#) , [\\_\\_msg\\_bad\\_inc](#) , [\\_\\_msg\\_bad\\_dec](#) ,  
[\\_\\_msg\\_iter\\_subscript\\_oob](#) , [\\_\\_msg\\_advance\\_oob](#) , [\\_\\_msg\\_retreat\\_oob](#) , [\\_\\_msg\\_iter\\_compare\\_bad](#) ,  
[\\_\\_msg\\_compare\\_different](#) , [\\_\\_msg\\_iter\\_order\\_bad](#) , [\\_\\_msg\\_order\\_different](#) , [\\_\\_msg\\_distance\\_bad](#) ,  
[\\_\\_msg\\_distance\\_different](#) , [\\_\\_msg\\_deref\\_istream](#) , [\\_\\_msg\\_inc\\_istream](#) , [\\_\\_msg\\_output\\_ostream](#) ,  
[\\_\\_msg\\_deref\\_istreambuf](#) , [\\_\\_msg\\_inc\\_istreambuf](#) , [\\_\\_msg\\_insert\\_after\\_end](#) , [\\_\\_msg\\_erase\\_after\\_bad](#) ,  
[\\_\\_msg\\_valid\\_range2](#) , [\\_\\_msg\\_local\\_iter\\_compare\\_bad](#) , [\\_\\_msg\\_non\\_empty\\_range](#) , [\\_\\_msg\\_self\\_move](#)↵  
[\\_\\_assign](#) ,  
[\\_\\_msg\\_bucket\\_index\\_oob](#) , [\\_\\_msg\\_valid\\_load\\_factor](#) , [\\_\\_msg\\_equal\\_allocs](#) , [\\_\\_msg\\_insert\\_range](#)↵  
[from\\_self](#) ,  
[\\_\\_msg\\_irreflexive\\_ordering](#) }
- enum [\\_Distance\\_precision](#) {  
[\\_\\_dp\\_none](#) , [\\_\\_dp\\_equality](#) , [\\_\\_dp\\_sign](#) , [\\_\\_dp\\_sign\\_max\\_size](#) ,  
[\\_\\_dp\\_exact](#) }

## Functions

- template<typename [\\_Iterator](#)>  
constexpr [\\_Iterator](#) [\\_\\_base](#) ([\\_Iterator](#) \_\_it)
- template<typename [\\_Iterator](#), typename [\\_Sequence](#)>  
[\\_Iterator](#) [\\_\\_base](#) (const [\\_Safe\\_iterator](#)< [\\_Iterator](#), [\\_Sequence](#), [std::random\\_access\\_iterator\\_tag](#) > &\_\_it)
- template<typename [\\_Iterator](#)>  
constexpr auto [\\_\\_base](#) (const [std::move\\_iterator](#)< [\\_Iterator](#) > &\_\_it) -> decltype(std::make\_move\_iterator(\_\_↵  
base(\_\_it.base())))

- `template<typename _Iterator, typename _Sequence>`  
`constexpr std::reverse_iterator< _Iterator > __base (const std::reverse_iterator< _Safe_iterator< _Iterator, _↵`  
`Sequence, std::random_access_iterator_tag > > &__it)`
- `template<typename _InputIterator, typename _Size>`  
`constexpr bool __can_advance (_InputIterator, _Size)`
- `template<typename _InputIterator, typename _Diff>`  
`constexpr bool __can_advance (_InputIterator, const std::pair< _Diff, _Distance_precision > &, int)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>`  
`bool __can_advance (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _Size)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>`  
`bool __can_advance (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const std::pair< _Diff,`  
`_Distance_precision > &, int)`
- `template<typename _Iterator, typename _Size>`  
`constexpr bool __can_advance (const std::move_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator, typename _Diff>`  
`constexpr bool __can_advance (const std::move_iterator< _Iterator > &__it, const std::pair< _Diff,`  
`_Distance_precision > &__dist, int __way)`
- `template<typename _Iterator, typename _Size>`  
`constexpr bool __can_advance (const std::reverse_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator, typename _Diff>`  
`constexpr bool __can_advance (const std::reverse_iterator< _Iterator > &__it, const std::pair< _Diff,`  
`_Distance_precision > &__dist, int __way)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool __check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↵`  
`value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred>`  
`constexpr bool __check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↵`  
`value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↵`  
`value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred>`  
`constexpr bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↵`  
`value, _Pred __pred)`
- `template<typename _Iterator>`  
`constexpr bool __check_singular (_Iterator const &)`
- `template<typename _Tp>`  
`constexpr bool __check_singular (_Tp *const &__ptr)`
- `bool __check_singular_aux (const _Safe_iterator_base * __x)`
- `bool __check_singular_aux (const class _Safe_iterator_base *)`
- `bool __check_singular_aux (const void *)`
- `template<typename _InputIterator>`  
`constexpr bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred,`  
`std::forward_iterator_tag)`
- `template<typename _ForwardIterator>`  
`constexpr bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)`

- `template<typename _InputIterator>`  
`constexpr bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate>`  
`constexpr bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::false\_type)`
- `template<typename _InputIterator>`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::false\_type)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::true\_type)`
- `template<typename _InputIterator>`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::true\_type)`
- `template<typename _CharT, typename _Integer>`  
`const _CharT * __check_string (const _CharT * __s, _Integer __n, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _CharT>`  
`const _CharT * __check_string (const _CharT * __s, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _InputIterator>`  
`_InputIterator __check_valid_range (const _InputIterator &__first, const _InputIterator &__last, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __foreign_iterator (const Safe\_iterator< _Iterator, _Sequence, _Category > &__it, _InputIterator __other, _InputIterator __other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Integral>`  
`bool __foreign_iterator_aux (const Safe\_iterator< _Iterator, _Sequence, _Category > &, _Integral, _Integral, std::true\_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __foreign_iterator_aux (const Safe\_iterator< _Iterator, _Sequence, _Category > &__it, _InputIterator __other, _InputIterator __other_end, std::false\_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, typename _OtherCategory>`  
`bool __foreign_iterator_aux2 (const Safe\_iterator< _Iterator, _Sequence, _Category > &, const Safe\_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &, const Safe\_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __foreign_iterator_aux2 (const Safe\_iterator< _Iterator, _Sequence, _Category > &__it, const _InputIterator &__other, const _InputIterator &__other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>`  
`bool __foreign_iterator_aux2 (const Safe\_iterator< _Iterator, _Sequence, _Category > &__it, const Safe\_iterator< _OtherIterator, _Sequence, _Category > &__other, const Safe\_iterator< _OtherIterator, _Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __foreign_iterator_aux3 (const Safe\_iterator< _Iterator, _Sequence, _Category > &, const _InputIterator &, const _InputIterator &, std::false\_type)`

- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __foreign_iterator_aux3 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _InputIterator &__other, const _InputIterator &__other_end, std::__true_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool __foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence, _Category > &,...)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool __foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it, const typename _Sequence::value_type *__other)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __get_distance ( _Iterator __lhs, _Iterator __rhs)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __get_distance ( _Iterator __lhs, _Iterator __rhs, std::input_iterator_tag)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __get_distance ( _Iterator __lhs, _Iterator __rhs, std::random_access_iterator_tag)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __get_distance (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __get_distance (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _Iterator > &__last)`
- `template<typename _Iterator>`  
`constexpr bool __is_irreflexive ( _Iterator __it)`
- `template<typename _Iterator, typename _Pred>`  
`constexpr bool __is_irreflexive_pred ( _Iterator __it, _Pred __pred)`
- `template<typename _Iterator>`  
`constexpr _Iterator __unsafe ( _Iterator __it)`
- `template<typename _Iterator, typename _Sequence>`  
`_Iterator __unsafe (const __Safe_iterator< _Iterator, _Sequence > &__it)`
- `template<typename _Iterator, typename _UContainer>`  
`_Iterator __unsafe (const __Safe_local_iterator< _Iterator, _UContainer > &__it)`
- `template<typename _Iterator>`  
`constexpr auto __unsafe (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__unsafe(__it.base())))`
- `template<typename _Iterator>`  
`constexpr auto __unsafe (const std::reverse_iterator< _Iterator > &__it) -> decltype(std::__make_reverse_iterator(__unsafe(__it.base())))`
- `template<typename _InputIterator>`  
`constexpr bool __valid_range ( _InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`constexpr bool __valid_range ( _InputIterator __first, _InputIterator __last, typename _Distance_traits< _InputIterator >::__type &__dist)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool __valid_range (const __Safe_iterator< _Iterator, _Sequence, _Category > &, const __Safe_iterator< _InputIterator, _Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool __valid_range (const __Safe_iterator< _Iterator, _Sequence, _Category > &, const __Safe_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _Iterator, typename _Sequence>`  
`bool __valid_range (const __Safe_local_iterator< _Iterator, _Sequence > &, const __Safe_local_iterator< _InputIterator, _Sequence > &)`
- `template<typename _Iterator, typename _Sequence>`  
`bool __valid_range (const __Safe_local_iterator< _Iterator, _Sequence > &, const __Safe_local_iterator< _InputIterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)`

- `template<typename _Iterator, typename _UContainer>`  
`bool __valid_range (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last)`
- `template<typename _Iterator, typename _UContainer>`  
`bool __valid_range (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last, typename _Distance_traits< _Iterator >::__type &__dist_info)`
- `template<typename _Iterator>`  
`constexpr bool __valid_range (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _Iterator>`  
`constexpr bool __valid_range (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _InputIterator>`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, std::__false_type)`
- `template<typename _InputIterator>`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _InputIterator>`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, std::random_access_iterator_tag)`
- `template<typename _InputIterator>`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, typename _Distance_traits< _InputIterator >::__type &__dist, std::__false_type)`
- `template<typename _Integral>`  
`constexpr bool __valid_range_aux (_Integral, _Integral, std::__true_type)`
- `template<typename _Integral>`  
`constexpr bool __valid_range_aux (_Integral, _Integral, typename _Distance_traits< _Integral >::__type &__dist, std::__true_type)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic_istream< _CharT, _Traits > &getline (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic_istream< _CharT, _Traits > &getline (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator< (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const basic_string< _CharT, _Traits, _Allocator > & __str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator<= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator== (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator> (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator>= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator>= (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator>= (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Allocator > & __str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`void swap (basic_string< _CharT, _Traits, _Allocator > & __lhs, basic_string< _CharT, _Traits, _Allocator > & __rhs)`

#### 4.4.1 Detailed Description

GNU debug classes for public use.

#### 4.4.2 Typedef Documentation

##### `u16string`

```
typedef basic_string<char16_t> __gnu_debug::u16string
```

A string of `char16_t`.

## u32string

typedef [basic\\_string](#)<char32\_t> [\\_\\_gnu\\_debug::u32string](#)  
A string of char32\_t.

### 4.4.3 Enumeration Type Documentation

#### [\\_\\_Distance\\_precision](#)

enum [\\_\\_gnu\\_debug::\\_\\_Distance\\_precision](#)  
The precision to which we can calculate the distance between two iterators.

### 4.4.4 Function Documentation

#### [\\_\\_base\(\)](#)

```
template<typename _Iterator>
_Iterator __gnu_debug::__base (
    _Iterator __it) [inline], [constexpr]
```

Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the [\\_\\_valid\\_range](#) function thanks to the < operator.  
Referenced by [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_\\_M\\_before\\_dereferenceable\(\)](#).

#### [\\_\\_check\\_singular\(\)](#)

```
template<typename _Tp>
bool __gnu_debug::__check_singular (
    _Tp *const & __ptr) [inline], [constexpr]
```

Non-NULL pointers are nonsingular.

#### [\\_\\_check\\_singular\\_aux\(\)](#)

```
bool __gnu_debug::__check_singular_aux (
    const \_Safe\_iterator\_base * __x) [inline]
```

Iterators that derive from [\\_Safe\\_iterator\\_base](#) can be determined singular or non-singular.  
References [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator\\_base::\\_\\_M\\_singular\(\)](#).

#### [\\_\\_check\\_string\(\)](#) [1/2]

```
template<typename _CharT, typename _Integer>
const _CharT * __gnu_debug::__check_string (
    const _CharT * __s,
    _Integer __n,
    const char * __file,
    unsigned int __line,
    const char * __function) [inline]
```

Checks that [\\_\\_s](#) is non-NULL or [\\_\\_n](#) == 0, and then returns [\\_\\_s](#).

#### [\\_\\_check\\_string\(\)](#) [2/2]

```
template<typename _CharT>
const _CharT * __gnu_debug::__check_string (
    const _CharT * __s,
    const char * __file,
```

```
    unsigned int __line,
    const char * __function) [inline]
```

Checks that `__s` is non-NULL and then returns `__s`.

#### `__foreign_iterator_aux2()` [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator,
typename _OtherSequence, typename _OtherCategory>
bool __gnu_debug::__foreign_iterator_aux2 (
    const _Safe_iterator< _Iterator, _Sequence, _Category > & ,
    const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > & ,
    const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > & ) [inline]
```

Handle debug iterators from different types of container.

#### `__foreign_iterator_aux2()` [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>
bool __gnu_debug::__foreign_iterator_aux2 (
    const _Safe_iterator< _Iterator, _Sequence, _Category > & __it,
    const _Safe_iterator< _OtherIterator, _Sequence, _Category > & __other,
    const _Safe_iterator< _OtherIterator, _Sequence, _Category > & ) [inline]
```

Handle debug iterators from the same type of container.

#### `__get_distance()`

```
template<typename _Iterator>
_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (
    _Iterator __lhs,
    _Iterator __rhs,
    std::random_access_iterator_tag ) [inline], [constexpr]
```

Determine the distance between two iterators with some known precision.

References [std::make\\_pair\(\)](#).

#### `__valid_range()` [1/3]

```
template<typename _InputIterator>
bool __gnu_debug::__valid_range (
    _InputIterator __first,
    _InputIterator __last,
    typename _Distance_traits< _InputIterator >::__type & __dist) [inline], [constexpr]
```

Don't know what these iterators are, or if they are even iterators (we may get an integral type for `InputIterator`), so see if they are integral and pass them on to the next phase otherwise.

References [\\_\\_valid\\_range\\_aux\(\)](#).

#### `__valid_range()` [2/3]

```
template<typename _Iterator, typename _Sequence, typename _Category>
bool __gnu_debug::__valid_range (
    const _Safe_iterator< _Iterator, _Sequence, _Category > & __first,
    const _Safe_iterator< _Iterator, _Sequence, _Category > & __last,
    typename _Distance_traits< _Iterator >::__type & __dist) [inline]
```

Safe iterators know how to check if they form a valid range.



**\_\_valid\_range()** [3/3]

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::__valid_range (
    const __Safe_local_iterator< _Iterator, _UContainer > & __first,
    const __Safe_local_iterator< _Iterator, _UContainer > & __last,
    typename _Distance_traits< _Iterator >::__type & __dist_info) [inline]
```

Safe local iterators know how to check if they form a valid range.

**\_\_valid\_range\_aux()** [1/2]

```
template<typename _InputIterator>
bool __gnu_debug::__valid_range_aux (
    _InputIterator __first,
    _InputIterator __last,
    std::__false_type ) [inline], [constexpr]
```

We have iterators, so figure out what kind of iterators they are to see if we can check the range ahead of time.

References [std::\\_\\_iterator\\_category\(\)](#), and [\\_\\_valid\\_range\\_aux\(\)](#).

**\_\_valid\_range\_aux()** [2/2]

```
template<typename _Integral>
bool __gnu_debug::__valid_range_aux (
    _Integral ,
    _Integral ,
    std::__true_type ) [inline], [constexpr]
```

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Referenced by [\\_\\_valid\\_range\(\)](#), and [\\_\\_valid\\_range\\_aux\(\)](#).

**4.5 \_\_gnu\_internal Namespace Reference****4.5.1 Detailed Description**

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

**4.6 \_\_gnu\_parallel Namespace Reference****Classes**

- struct [\\_\\_accumulate\\_binop\\_reduct](#)
- struct [\\_\\_accumulate\\_selector](#)
- struct [\\_\\_adjacent\\_difference\\_selector](#)
- struct [\\_\\_adjacent\\_find\\_selector](#)
- class [\\_\\_binder1st](#)
- class [\\_\\_binder2nd](#)
- struct [\\_\\_count\\_if\\_selector](#)
- struct [\\_\\_count\\_selector](#)
- struct [\\_\\_fill\\_selector](#)
- struct [\\_\\_find\\_first\\_of\\_selector](#)
- struct [\\_\\_find\\_if\\_selector](#)
- struct [\\_\\_for\\_each\\_selector](#)
- struct [\\_\\_generate\\_selector](#)
- struct [\\_\\_generic\\_find\\_selector](#)

- struct `__generic_for_each_selector`
- struct `__identity_selector`
- struct `__inner_product_selector`
- struct `__max_element_reduct`
- struct `__min_element_reduct`
- struct `__mismatch_selector`
- struct `__multiway_merge_3_variant_sentinel_switch`
- struct `__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- struct `__multiway_merge_4_variant_sentinel_switch`
- struct `__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- struct `__multiway_merge_k_variant_sentinel_switch`
- struct `__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- struct `__replace_if_selector`
- struct `__replace_selector`
- struct `__transform1_selector`
- struct `__transform2_selector`
- class `__unary_negate`
- struct `_DRandomShufflingGlobalData`
- struct `_DRSSorterPU`
- struct `_DummyReduct`
- class `_EqualFromLess`
- struct `_EqualTo`
- class `_GuardedIterator`
- class `_IteratorPair`
- class `_IteratorTriple`
- struct `_Job`
- struct `_Less`
- class `_Lexicographic`
- class `_LexicographicReverse`
- class `_LoserTree`
- class `_LoserTree< false, _Tp, _Compare >`
- class `_LoserTreeBase`
- class `_LoserTreePointer`
- class `_LoserTreePointer< false, _Tp, _Compare >`
- class `_LoserTreePointerBase`
- class `_LoserTreePointerUnguarded`
- class `_LoserTreePointerUnguarded< false, _Tp, _Compare >`
- class `_LoserTreePointerUnguardedBase`
- struct `_LoserTreeTraits`
- class `_LoserTreeUnguarded`
- class `_LoserTreeUnguarded< false, _Tp, _Compare >`
- class `_LoserTreeUnguardedBase`
- struct `_Multiplies`
- struct `_Nothing`
- struct `_Piece`
- struct `_Plus`
- struct `_PMWMSSortingData`
- class `_PseudoSequence`
- class `_PseudoSequenceIterator`
- struct `_QSBThreadLocal`
- class `_RandomNumber`

- class `_RestrictedBoundedConcurrentQueue`
- struct `_SamplingSorter`
- struct `_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`
- struct `_Settings`
- struct `_SplitConsistently`
- struct `_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >`
- struct `_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >`
- struct `balanced_quicksort_tag`
- struct `balanced_tag`
- struct `constant_size_blocks_tag`
- struct `default_parallel_tag`
- struct `equal_split_tag`
- struct `exact_tag`
- struct `find_tag`
- struct `growing_blocks_tag`
- struct `multiway_mergesort_exact_tag`
- struct `multiway_mergesort_sampling_tag`
- struct `multiway_mergesort_tag`
- struct `omp_loop_static_tag`
- struct `omp_loop_tag`
- struct `parallel_tag`
- struct `quicksort_tag`
- struct `sampling_tag`
- struct `sequential_tag`
- struct `unbalanced_tag`

## Typedefs

- typedef unsigned short `_BinIndex`
- typedef int64\_t `_CASable`
- typedef uint64\_t `_SequenceIndex`
- typedef uint16\_t `_ThreadIndex`

## Enumerations

- enum `_AlgorithmStrategy` { `heuristic` , `force_sequential` , `force_parallel` }
- enum `_FindAlgorithm` { `GROWING_BLOCKS` , `CONSTANT_SIZE_BLOCKS` , `EQUAL_SPLIT` }
- enum `_MultiwayMergeAlgorithm` { `LOSER_TREE` }
- enum `_Parallelism` {  
    `sequential` , `parallel_unbalanced` , `parallel_balanced` , `parallel_omp_loop` ,  
    `parallel_omp_loop_static` , `parallel_taskqueue` }
- enum `_PartialSumAlgorithm` { `RECURSIVE` , `LINEAR` }
- enum `_SortAlgorithm` { `MWMS` , `QS` , `QS_BALANCED` }
- enum `_SplittingAlgorithm` { `SAMPLING` , `EXACT` }

## Functions

- `template<typename _Tp>`  
`_Tp __add_omp (volatile _Tp *__ptr, _Tp __addend)`
- `template<typename _RAIter, typename _DifferenceTp>`  
`void __calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)`
- `template<typename _Tp>`  
`bool __cas_omp (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)`
- `template<typename _Tp>`  
`bool __compare_and_swap (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator __copy_tail (std::pair< _Iter, _Iter > __b, std::pair< _Iter, _Iter > __e, _OutputIterator __r)`
- `void __decode2 (_CASable __x, int &__a, int &__b)`
- `template<typename _RAIter, typename _DifferenceTp>`  
`void __determine_samples (_PMWMSortingData< _RAIter > *__sd, _DifferenceTp __num_samples)`
- `_CASable __encode2 (int __a, int __b)`
- `template<typename _DifferenceType, typename _OutputIterator>`  
`_OutputIterator __equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)`
- `template<typename _DifferenceType>`  
`_DifferenceType __equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)`
- `template<typename _Tp>`  
`_Tp __fetch_and_add (volatile _Tp *__ptr, _Tp __addend)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
`std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
`std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
`std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
`std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)`
- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>`  
`_UserOp __for_each_template_random_access (_Iter __begin, _Iter __end, _UserOp __user_op, _Functionality & __functionality, _Red __reduction, _Result __reduction_start, _Result & __output, typename std::iterator_traits< _Iter >::difference_type __bound, _Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op __for_each_template_random_access_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op __for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op __for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op __for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`

- `_ThreadIndex __get_max_threads ()`
- `bool __is_parallel (const _Parallelism __p)`
- `template<typename _Iter, typename _Compare>`  
`bool __is_sorted (_Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAlter, typename _Compare>`  
`_RAlter __median_of_three_iterators (_RAlter __a, _RAlter __b, _RAlter __c, _Compare __comp)`
- `template<typename _RAlter1, typename _RAlter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>`  
`_OutputIterator __merge_advance (_RAlter1 &__begin1, _RAlter1 __end1, _RAlter2 &__begin2, _RAlter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAlter1, typename _RAlter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>`  
`_OutputIterator __merge_advance_movc (_RAlter1 &__begin1, _RAlter1 __end1, _RAlter2 &__begin2, _RAlter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAlter1, typename _RAlter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>`  
`_OutputIterator __merge_advance_usual (_RAlter1 &__begin1, _RAlter1 __end1, _RAlter2 &__begin2, _RAlter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAlter1, typename _RAlter3, typename _Compare>`  
`_RAlter3 __parallel_merge_advance (_RAlter1 &__begin1, _RAlter1 __end1, _RAlter1 &__begin2, _RAlter1 __end2, _RAlter3 __target, typename std::iterator_traits<_RAlter1>::difference_type __max_length, _Compare __comp)`
- `template<typename _RAlter1, typename _RAlter2, typename _RAlter3, typename _Compare>`  
`_RAlter3 __parallel_merge_advance (_RAlter1 &__begin1, _RAlter1 __end1, _RAlter2 &__begin2, _RAlter2 __end2, _RAlter3 __target, typename std::iterator_traits<_RAlter1>::difference_type __max_length, _Compare __comp)`
- `template<typename _RAlter, typename _Compare>`  
`void __parallel_nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp)`
- `template<typename _RAlter, typename _Compare>`  
`void __parallel_partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator __parallel_partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator __parallel_partial_sum_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter>::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator __parallel_partial_sum_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter>::difference_type __n)`
- `template<typename _RAlter, typename _Predicate>`  
`std::iterator_traits<_RAlter>::difference_type __parallel_partition (_RAlter __begin, _RAlter __end, _Predicate __pred, _ThreadIndex __num_threads)`
- `template<typename _RAlter, typename _RandomNumberGenerator>`  
`void __parallel_random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rng, _RandomNumber())`
- `template<typename _RAlter, typename _RandomNumberGenerator>`  
`void __parallel_random_shuffle_drs (_RAlter __begin, _RAlter __end, typename std::iterator_traits<_RAlter>::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAlter, typename _RandomNumberGenerator>`  
`void __parallel_random_shuffle_drs_pu (_DRSSorterPU<_RAlter, _RandomNumberGenerator> *__pus)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __parallel_set_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __parallel_set_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`

- `template<typename _Iter, typename _OutputIterator, typename _Operation>`  
`_OutputIterator __parallel_set_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _↵`  
`OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __parallel_set_symmetric_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter ↵`  
`__end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __parallel_set_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _Output↵`  
`Iterator __result, _Compare __comp)`
- `template<bool __stable, typename _RAIter, typename _Compare, typename \_Parallelism>`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced\_quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default\_parallel\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_exact\_tag __↵`  
`parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_sampling\_tag ↵`  
`__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_tag __↵`  
`parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare>`  
`void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num↵`  
`__threads)`
- `template<typename _RAIter, typename _Compare>`  
`std::iterator\_traits< _RAIter >::difference_type __parallel_sort_qs_divide (_RAIter __begin, _RAIter __↵`  
`end, _Compare __comp, typename std::iterator\_traits< _RAIter >::difference_type __pivot_rank, typename`  
`std::iterator\_traits< _RAIter >::difference_type __num_samples, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator>`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate>`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate ↵`  
`__binary_pred)`
- `template<typename _RAIter, typename _Compare>`  
`void __qsb_conquer (\_QSBThreadLocal< _RAIter > *__tls, _RAIter __begin, _RAIter __end, _Compare __↵`  
`comp, \_ThreadIndex __iam, \_ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare>`  
`std::iterator\_traits< _RAIter >::difference_type __qsb_divide (_RAIter __begin, _RAIter __end, _Compare __↵`  
`comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`void __qsb_local_sort_with_helping (\_QSBThreadLocal< _RAIter > *__tls, _Compare &__comp, \_ThreadIndex ↵`  
`__iam, bool __wait)`

- `template<typename _RandomNumberGenerator>`  
`int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Size>`  
`_Size __rd_log2 (_Size __n)`
- `template<typename _Tp>`  
`_Tp __round_up_to_pow2 (_Tp __x)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred>`  
`__RAIter1 __search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type <←  
::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void __sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)`
- `template<typename _Iter>`  
`void __shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter>`  
`void __shrink_and_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `void __yield ()`
- `template<typename _Iter, typename _FunctorType>`  
`size_t list_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __←  
num_parts, _FunctorType & __f, int __oversampling=0)`
- `template<typename _Tp>`  
`const _Tp & max (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp>`  
`const _Tp & min (const _Tp & __a, const _Tp & __b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>`  
`void multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator <←  
__begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< <←  
_RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>`  
`_Tp multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType & <←  
__offset, _Compare __comp=std::less< _Tp >())`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <←  
__target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <←  
__target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <←  
__target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <←  
__target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <←  
__target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _<←  
DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __<←  
target, _DifferenceTp __length, _Compare __comp)`

- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_4_variant ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterlterator, typename _Compare, typename _DifferenceType>`  
`void multiway_merge_exact_splitting ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _LT, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_loser_tree ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename _UnguardedLoserTree, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_loser_tree_sentinel ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterlterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_loser_tree_unguarded ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterlterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterlterator, typename _Compare, typename _DifferenceType>`  
`void multiway_merge_sampling_splitting ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _RAIterPairlterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels ( _RAIterPairlterator __seqs_begin, _RAIterPairlterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairlterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels ( _RAIterPairlterator __seqs_begin, _RAIterPairlterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairlterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels ( _RAIterPairlterator __seqs_begin, _RAIterPairlterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairlterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels ( _RAIterPairlterator __seqs_begin, _RAIterPairlterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairlterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels ( _RAIterPairlterator __seqs_begin, _RAIterPairlterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, typename _Compare>`  
`_RAIter3 parallel_multiway_merge ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare>`  
`void parallel_sort_mwms ( _RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare>`  
`void parallel_sort_mwms_pu ( PMWMSortingData< _RAIter > * __sd, _Compare & __comp)`
- `template<typename _RAIterPairlterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge ( _RAIterPairlterator __seqs_begin, _RAIterPairlterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairlterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge ( _RAIterPairlterator __seqs_begin, _RAIterPairlterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`



- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`

## Variables

- static const int [\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_CASable\\_mask](#)

### 4.6.1 Detailed Description

GNU parallel code for public use.

### 4.6.2 Typedef Documentation

#### [\\_BinIndex](#)

```
typedef unsigned short \_\_gnu\_parallel::\_BinIndex
```

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

#### [\\_CASable](#)

```
typedef int64_t \_\_gnu\_parallel::\_CASable
```

Longest compare-and-swappable integer type on this platform.

#### [\\_SequenceIndex](#)

```
typedef uint64_t \_\_gnu\_parallel::\_SequenceIndex
```

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

#### [\\_ThreadIndex](#)

```
typedef uint16_t \_\_gnu\_parallel::\_ThreadIndex
```

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

### 4.6.3 Enumeration Type Documentation

#### `_AlgorithmStrategy`

enum `__gnu_parallel::_AlgorithmStrategy`

Strategies for run-time algorithm selection:

#### `_FindAlgorithm`

enum `__gnu_parallel::_FindAlgorithm`

Find algorithms:

#### `_MultiwayMergeAlgorithm`

enum `__gnu_parallel::_MultiwayMergeAlgorithm`

Merging algorithms:

#### `_Parallelism`

enum `__gnu_parallel::_Parallelism`

Run-time equivalents for the compile-time tags.

Enumerator

<code>sequential</code>	Not parallel.
<code>parallel_unbalanced</code>	Parallel unbalanced (equal-sized chunks).
<code>parallel_balanced</code>	Parallel balanced (work-stealing).
<code>parallel_omp_loop</code>	Parallel with OpenMP dynamic load-balancing.
<code>parallel_omp_loop_static</code>	Parallel with OpenMP static load-balancing.
<code>parallel_taskqueue</code>	Parallel with OpenMP taskqueue construct.

#### `_PartialSumAlgorithm`

enum `__gnu_parallel::_PartialSumAlgorithm`

Partial sum algorithms: recursive, linear.

#### `_SortAlgorithm`

enum `__gnu_parallel::_SortAlgorithm`

Sorting algorithms:

#### `_SplittingAlgorithm`

enum `__gnu_parallel::_SplittingAlgorithm`

Sorting/merging algorithms: sampling, `__exact`.

### 4.6.4 Function Documentation

#### `__calc_borders()`

```
template<typename _RAIter, typename _DifferenceTp>
void __gnu_parallel::__calc_borders (
    _RAIter __elements,
```

```

        _DifferenceTp __length,
        _DifferenceTp * __off)

```

Precalculate `__advances` for Knuth-Morris-Pratt algorithm.

#### Parameters

<code>__elements</code>	Begin iterator of sequence to search for.
<code>__length</code>	Length of sequence to search for.
<code>__off</code>	Returned <code>__offsets</code> .

Referenced by [\\_\\_search\\_template\(\)](#).

#### `__compare_and_swap()`

```

template<typename _Tp>
bool __gnu_parallel::__compare_and_swap (
    volatile _Tp * __ptr,
    _Tp __comparand,
    _Tp __replacement) [inline]

```

Compare-and-swap.

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return true, return false otherwise.

#### Parameters

<code>__ptr</code>	Pointer to signed integer.
<code>__comparand</code>	Compare value.
<code>__replacement</code>	Replacement value.

Referenced by [\\_\\_parallel\\_partition\(\)](#), [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>::pop\\_back\(\)](#), and [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>::pop\\_front\(\)](#).

#### `__decode2()`

```

void __gnu_parallel::__decode2 (
    _CASable __x,
    int & __a,
    int & __b) [inline]

```

Decode two integers from one `gnu_parallel::_CASable`.

#### Parameters

<code>↔ __x</code>	<code>__gnu_parallel::_CASable</code> to decode integers from.
<code>↔ __a</code>	First integer, to be decoded from the most-significant <code>_CASable_bits/2</code> bits of <code>__x</code> .
<code>↔ __b</code>	Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits of <code>__x</code> .

#### See also

`__encode2`

References [\\_CASable\\_bits](#), and [\\_CASable\\_mask](#).

Referenced by [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>::pop\\_back\(\)](#), [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>::pop\\_front\(\)](#), and [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>::push\\_front\(\)](#).

**`__determine_samples()`**

```
template<typename _RAIter, typename _DifferenceTp>
void __gnu_parallel::__determine_samples (
    _PMWSSortingData< _RAIter > * __sd,
    _DifferenceTp __num_samples)
```

Select `_M_samples` from a sequence.

**Parameters**

<code>__sd</code>	Pointer to algorithm data. Result will be placed in <code>__sd-&gt;_M_samples</code> .
<code>__num_samples</code>	Number of <code>_M_samples</code> to select.

References [\\_\\_equally\\_split\(\)](#), [\\_\\_gnu\\_parallel::PMWSSortingData< \\_RAIter >::\\_M\\_samples](#), [\\_\\_gnu\\_parallel::PMWSSortingData< \\_RAIter >::\\_M\\_starts](#) and [\\_\\_gnu\\_parallel::PMWSSortingData< \\_RAIter >::\\_M\\_starts](#).

**`__encode2()`**

```
_CASable __gnu_parallel::__encode2 (
    int __a,
    int __b) [inline]
```

Encode two integers into one `gnu_parallel::_CASable`.

**Parameters**

<code>__a</code>	First integer, to be encoded in the most-significant <code>_CASable_bits/2</code> bits.
<code>__b</code>	Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits.

**Returns**

value encoding `__a` and `__b`.

**See also**

[\\_\\_decode2](#)

References [\\_CASable\\_bits](#).

Referenced by [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< \\_Tp >::\\_RestrictedBoundedConcurrentQueue\(\)](#), [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< \\_Tp >::pop\\_back\(\)](#), [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< \\_Tp >::push\\_front\(\)](#) and [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< \\_Tp >::push\\_front\(\)](#).

**`__equally_split()`**

```
template<typename _DifferenceType, typename _OutputIterator>
_OutputIterator __gnu_parallel::__equally_split (
    _DifferenceType __n,
    _ThreadIndex __num_threads,
    _OutputIterator __s)
```

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0,__n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

## Parameters

<code>__n</code>	Number of elements
<code>__num_threads</code>	Number of parts
<code>__s</code>	Splitters

## Returns

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Referenced by [\\_\\_determine\\_samples\(\)](#), [\\_\\_find\\_template\(\)](#), [\\_\\_parallel\\_partial\\_sum\\_linear\(\)](#), [\\_\\_parallel\\_unique\\_copy\(\)](#), [\\_\\_search\\_template\(\)](#), and [multiway\\_merge\\_exact\\_splitting\(\)](#).

**`__equally_split_point()`**

```
template<typename _DifferenceType>
_DifferenceType __gnu_parallel::__equally_split_point (
    _DifferenceType __n,
    _ThreadIndex __num_threads,
    _ThreadIndex __thread_no)
```

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

## Parameters

<code>__n</code>	Number of elements
<code>__num_threads</code>	Number of parts
<code>__thread_no</code>	Number of threads

## Returns

splitting point

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\\_ed\(\)](#).

**`__fetch_and_add()`**

```
template<typename _Tp>
_Tp __gnu_parallel::__fetch_and_add (
    volatile _Tp * __ptr,
    _Tp __addend) [inline]
```

Add a value to a variable, atomically.

## Parameters

<code>__ptr</code>	Pointer to a signed integer.
<code>__addend</code>	Value to add.

Referenced by [\\_\\_find\\_template\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#), [\\_\\_parallel\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_RestrictedBoundedConcurrentQueue<\\_Tp>::push\\_front\(\)](#).

**`__find_template()`** [1/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred,
    _Selector __selector) [inline]
```

Parallel `std::find`, switch for different algorithms.

**Parameters**

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	_Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

**Returns**

Place of finding in both sequences.

References [\\_\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), and [std::make\\_pair\(\)](#).

Referenced by [\\_\\_find\\_template\(\)](#).

**`__find_template()`** [2/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred,
    _Selector __selector,
    constant_size_blocks_tag )
```

Parallel `std::find`, constant block size variant.

**Parameters**

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	_Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

**Returns**

Place of finding in both sequences.

## See also

[\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#)

[\\_\\_gnu\\_parallel::\\_Settings::find\\_block\\_size](#) There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

References [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_initial\\_block\\_size](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#), [std::pair<\\_T1, \\_T2>::first](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), and [std::min\(\)](#).

**`__find_template()`** [3/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred,
    _Selector __selector,
    equal_split_tag )
```

Parallel `std::find`, equal splitting variant.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	<code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

## Returns

Place of finding in both sequences.

References [\\_\\_equally\\_split\(\)](#), and [\\_GLIBCXX\\_CALL](#).

**`__find_template()`** [4/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred,
    _Selector __selector,
    growing_blocks_tag )
```

Parallel `std::find`, growing block size variant.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
-----------------------	-----------------------------------

## Parameters

<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	<code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

## Returns

Place of finding in both sequences.

## See also

`__gnu_parallel::_Settings::find_sequential_search_size`  
`__gnu_parallel::_Settings::find_scale_factor`

There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

References [\\_\\_fetch\\_and\\_add\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_scale\\_factor](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#), [std::pair<\\_T1, \\_T2>::first](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::max\(\)](#), and [std::min\(\)](#).

`__for_each_template_random_access()`

```
template<typename _IIter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>
_Result __gnu_parallel::__for_each_template_random_access (
    _IIter __begin,
    _IIter __end,
    _UserOp __user_op,
    _Functionality & __functionality,
    _Red __reduction,
    _Result __reduction_start,
    _Result & __output,
    typename std::iterator_traits<_IIter>::difference_type __bound,
    _Parallelism __parallelism_tag)
```

Chose the desired algorithm by evaluating `__parallelism_tag`.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__user_op</code>	A user-specified functor (comparator, predicate, associative operator,...)
<code>__functionality</code>	functor to <i>process</i> an element with <code>__user_op</code> (depends on desired functionality, e. g. accumulate, <code>for_each</code> ,...)
<code>__reduction</code>	Reduction functor.
<code>__reduction_start</code>	Initial value for reduction.
<code>__output</code>	Output iterator.
<code>__bound</code>	Maximum number of elements processed.



<code>__parallelism_tag</code>	Parallelization method
--------------------------------	------------------------

References [\\_\\_for\\_each\\_template\\_random\\_access\\_ed\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\\_static\(\)](#), and [parallel\\_unbalanced\(\)](#).

### **`__for_each_template_random_access_ed()`**

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_ed (
    _RAIter __begin,
    _RAIter __end,
    _Op __o,
    _Fu & __f,
    _Red __r,
    _Result __base,
    _Result & __output,
    typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

#### Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, ...)
<code>__f</code>	Functor to "process" an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to "add" a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

#### Returns

User-supplied functor (that may contain a part of the result).

References [\\_\\_equally\\_split\\_point\(\)](#), and [min\(\)](#).

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\(\)](#).

### **`__for_each_template_random_access_omp_loop()`**

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_omp_loop (
    _RAIter __begin,
    _RAIter __end,
    _Op __o,
    _Fu & __f,
    _Red __r,
    _Result __base,
    _Result & __output,
    typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

## Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, etc.).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

## Returns

User-supplied functor (that may contain a part of the result).

References [min\(\)](#).

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\(\)](#).

**`__for_each_template_random_access_omp_loop_static()`**

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (
    _RAIter __begin,
    _RAIter __end,
    _Op __o,
    _Fu & __f,
    _Red __r,
    _Result __base,
    _Result & __output,
    typename std::iterator_traits< _RAIter >::difference_type __bound)

```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

## Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, ...).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed <code>__elements</code> (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

## Returns

User-supplied functor (that may contain a part of the result).

References [std::min\(\)](#).

**\_\_for\_each\_template\_random\_access\_workstealing()**

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
__Op __gnu_parallel::__for_each_template_random_access_workstealing (
    _RAIter __begin,
    _RAIter __end,
    _Op __op,
    _Fu & __f,
    _Red __r,
    _Result __base,
    _Result & __output,
    typename std::iterator_traits< _RAIter >::difference_type __bound)

```

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

**Parameters**

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__op</code>	User-supplied functor (comparator, predicate, adding functor, ...).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

**Returns**

User-supplied functor (that may contain a part of the result).

References [\\_\\_fetch\\_and\\_add\(\)](#), [\\_\\_yield\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Job<\\_DifferenceTp>::\\_M\\_first](#), [\\_\\_gnu\\_parallel::\\_Job<\\_DifferenceTp>::\\_M\\_last](#), [\\_\\_gnu\\_parallel::\\_Job<\\_DifferenceTp>::\\_M\\_load](#), [\\_\\_gnu\\_parallel::\\_Settings::cache\\_l](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [max\(\)](#), and [min\(\)](#).

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\(\)](#).

**\_\_is\_sorted()**

```
template<typename _IIter, typename _Compare>
bool __gnu_parallel::__is_sorted (
    _IIter __begin,
    _IIter __end,
    _Compare __comp)

```

Check whether [`__begin`, `__end`) is sorted according to `__comp`.

**Parameters**

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.

**Returns**

`true` if sorted, `false` otherwise.

Referenced by [\\_\\_sequential\\_multiway\\_merge\(\)](#), [multiway\\_merge\\_loser\\_tree\\_sentinel\(\)](#), and [parallel\\_multiway\\_merge\(\)](#).

**`__median_of_three_iterators()`**

```
template<typename _RAIter, typename _Compare>
_RAIter __gnu_parallel::__median_of_three_iterators (
    _RAIter __a,
    _RAIter __b,
    _RAIter __c,
    _Compare __comp)
```

Compute the median of three referenced elements, according to `__comp`.

**Parameters**

<code>__a</code>	First iterator.
<code>__b</code>	Second iterator.
<code>__c</code>	Third iterator.
<code>__comp</code>	Comparator.

Referenced by [\\_\\_qsb\\_divide\(\)](#).

**`__merge_advance()`**

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter2 & __begin2,
    _RAIter2 __end2,
    _OutputIterator __target,
    _DifferenceTp __max_length,
    _Compare __comp) [inline]
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

**Parameters**

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

**Returns**

Output end iterator.

References [\\_\\_merge\\_advance\\_movc\(\)](#), and [\\_GLIBCXX\\_CALL](#).

Referenced by [\\_\\_parallel\\_merge\\_advance\(\)](#), and [\\_\\_sequential\\_multiway\\_merge\(\)](#).

**\_\_merge\_advance\_movc()**

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance_movc (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter2 & __begin2,
    _RAIter2 __end2,
    _OutputIterator __target,
    _DifferenceTp __max_length,
    _Compare __comp)
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

**Parameters**

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

**Returns**

Output end iterator.

Referenced by [\\_\\_merge\\_advance\(\)](#).

**\_\_merge\_advance\_usual()**

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance_usual (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter2 & __begin2,
    _RAIter2 __end2,
    _OutputIterator __target,
    _DifferenceTp __max_length,
    _Compare __comp)
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

**Parameters**

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.

## Parameters

<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

## Returns

Output end iterator.

**`__parallel_merge_advance()`** [1/2]

```
template<typename _RAIter1, typename _RAIter3, typename _Compare>
_RAIter3 __gnu_parallel::__parallel_merge_advance (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter1 & __begin2,
    _RAIter1 __end2,
    _RAIter3 __target,
    typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
    _Compare __comp) [inline]
```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

## Returns

Output end iterator.

References [std::make\\_pair\(\)](#), [multiway\\_merge\\_exact\\_splitting\(\)](#), and [parallel\\_multiway\\_merge\(\)](#).

**`__parallel_merge_advance()`** [2/2]

```
template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>
_RAIter3 __gnu_parallel::__parallel_merge_advance (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter2 & __begin2,
    _RAIter2 __end2,
    _RAIter3 __target,
    typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
    _Compare __comp) [inline]
```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

## Returns

Output end iterator.

References [\\_\\_merge\\_advance\(\)](#).

**`__parallel_nth_element()`**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_nth_element (
    _RAIter __begin,
    _RAIter __nth,
    _RAIter __end,
    _Compare __comp)
```

Parallel implementation of `std::nth_element()`.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__nth</code>	Iterator of element that must be in position afterwards.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

References [\\_\\_parallel\\_partition\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::max\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::nth\\_element](#) and [\\_\\_gnu\\_parallel::\\_Settings::partition\\_minimal\\_n](#).

Referenced by [\\_\\_parallel\\_partial\\_sort\(\)](#).

**`__parallel_partial_sort()`**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_partial_sort (
    _RAIter __begin,
    _RAIter __middle,
    _RAIter __end,
    _Compare __comp)
```

Parallel implementation of `std::partial_sort()`.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__middle</code>	Sort until this position.
<code>__end</code>	End iterator of input sequence.

<code>__comp</code>	Comparator.
---------------------	-------------

References [\\_\\_parallel\\_nth\\_element\(\)](#).

### `__parallel_partial_sum()`

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum (
    _IIter __begin,
    _IIter __end,
    _OutputIterator __result,
    _BinaryOperation __bin_op)
```

Parallel partial sum front-`__end`.

#### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.

#### Returns

End iterator of output sequence.

References [\\_\\_parallel\\_partial\\_sum\\_linear\(\)](#), [\\_GLIBCXX\\_CALL](#), and [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#).

### `__parallel_partial_sum_basecase()`

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (
    _IIter __begin,
    _IIter __end,
    _OutputIterator __result,
    _BinaryOperation __bin_op,
    typename std::iterator_traits< _IIter >::value_type __value)
```

Base case prefix sum routine.

#### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.
<code>__value</code>	Start value. Must be passed since the neutral element is unknown in general.

#### Returns

End iterator of output sequence.

Referenced by [\\_\\_parallel\\_partial\\_sum\\_linear\(\)](#).



**\_\_parallel\_partial\_sum\_linear()**

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum_linear (
    _IIter __begin,
    _IIter __end,
    _OutputIterator __result,
    _BinaryOperation __bin_op,
    typename std::iterator_traits< _IIter >::difference_type __n)
```

Parallel partial sum implementation, two-phase approach, no recursion.

**Parameters**

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.
<code>__n</code>	Length of sequence.

**Returns**

End iterator of output sequence.

References [\\_\\_equally\\_split\(\)](#), [\\_\\_parallel\\_partial\\_sum\\_basecase\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::max\(\)](#), [std::min\(\)](#), and [\\_\\_gnu\\_parallel::\\_Settings::partial\\_sum\\_dilation](#).

Referenced by [\\_\\_parallel\\_partial\\_sum\(\)](#).

**\_\_parallel\_partition()**

```
template<typename _RAIter, typename _Predicate>
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_partition (
    _RAIter __begin,
    _RAIter __end,
    _Predicate __pred,
    _ThreadIndex __num_threads)
```

Parallel implementation of `std::partition`.

**Parameters**

<code>__begin</code>	Begin iterator of input sequence to split.
<code>__end</code>	End iterator of input sequence to split.
<code>__pred</code>	Partition predicate, possibly including some kind of pivot.
<code>__num_threads</code>	Maximum number of threads to use for this task.

**Returns**

Number of elements not fulfilling the predicate.

References [\\_\\_compare\\_and\\_swap\(\)](#), [\\_\\_fetch\\_and\\_add\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_GLIBCXX\\_VOLATILE](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::max\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::partition\\_chunk\\_share](#), and [\\_\\_gnu\\_parallel::\\_Settings::partition\\_chunk\\_size](#).

Referenced by [\\_\\_parallel\\_nth\\_element\(\)](#), [\\_\\_parallel\\_sort\\_qs\\_divide\(\)](#), and [\\_\\_qsb\\_divide\(\)](#).

**`__parallel_random_shuffle()`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle (
    _RAIter __begin,
    _RAIter __end,
    _RandomNumberGenerator __rng = _RandomNumber()) [inline]
```

Parallel random public call.

**Parameters**

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__rng</code>	Random number generator to use.

References [\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#).

**`__parallel_random_shuffle_drs()`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle_drs (
    _RAIter __begin,
    _RAIter __end,
    typename std::iterator_traits< _RAIter >::difference_type __n,
    _ThreadIndex __num_threads,
    _RandomNumberGenerator & __rng)
```

Main parallel random shuffle step.

**Parameters**

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__n</code>	Length of sequence.
<code>__num_threads</code>	Number of threads to use.
<code>__rng</code>	Random number generator to use.

References [\\_\\_gnu\\_parallel::\\_\\_DRSSorterPU< \\_RAIter, \\_RandomNumberGenerator >::\\_\\_bins\\_end](#), [\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#), [\\_\\_rd\\_log2\(\)](#), [\\_\\_round\\_up\\_to\\_pow2\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), [\\_\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_\\_DRandomShufflingGlobalData](#), [\\_\\_gnu\\_parallel::\\_\\_DRSSorterPU< \\_RAIter, \\_RandomNumberGenerator >::\\_\\_M\\_bins\\_begin](#), [\\_\\_gnu\\_parallel::\\_\\_DRandomShufflingGlobalData](#), [\\_\\_gnu\\_parallel::\\_\\_DRandomShufflingGlobalData< \\_RAIter >::\\_\\_M\\_num\\_bins](#), [\\_\\_gnu\\_parallel::\\_\\_DRandomShufflingGlobalData< \\_RAIter >::\\_\\_M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::\\_\\_DRSSorterPU< \\_RAIter, \\_RandomNumberGenerator >::\\_\\_M\\_seed](#), [\\_\\_gnu\\_parallel::\\_\\_DRandomShufflingGlobalData< \\_RAIter >::\\_\\_M\\_temporaries](#), [\\_\\_gnu\\_parallel::\\_\\_Settings::get\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_Settings::L2\\_cache\\_size](#), [std::max\(\)](#), [std::numeric\\_limits< \\_Tp >::max\(\)](#), [std::min\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_Settings::TLB](#).

Referenced by [\\_\\_parallel\\_random\\_shuffle\(\)](#).

**`__parallel_random_shuffle_drs_pu()`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle_drs_pu (
    _DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus)
```

Random shuffle code executed by each thread.

## Parameters

<code>__pus</code>	Array of thread-local data records.
--------------------	-------------------------------------

References [\\_\\_random\\_number\\_pow2\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter>::M](#), [\\_\\_gnu\\_parallel::\\_DRSSorterPU<\\_RAIter, \\_RandomNumberGenerator>::M\\_bins\\_begin](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter>::M\\_num\\_bins](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter>](#), [\\_\\_gnu\\_parallel::\\_DRSSorterPU<\\_RAIter, \\_RandomNumberGenerator>::M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::\\_DRSSorterPU<\\_RAIter,](#), [\\_\\_gnu\\_parallel::\\_DRSSorterPU<\\_RAIter, \\_RandomNumberGenerator>::M\\_seed](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter>::M\\_starts](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter>::M](#) and [std::partial\\_sum\(\)](#).

Referenced by [\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#).

**`__parallel_sort()`** [1/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    balanced_quicksort_tag __parallelism) [inline]
```

Choose balanced quicksort for parallel sorting.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

References [\\_\\_gnu\\_parallel::parallel\\_tag::get\\_num\\_threads\(\)](#), [\\_\\_parallel\\_sort\\_qsb\(\)](#), and [\\_GLIBCXX\\_CALL](#).

Here is the call graph for this function:

**`__parallel_sort()`** [2/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
```

```

    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    default_parallel_tag __parallelism) [inline]

```

Choose multiway mergesort with exact splitting, for parallel sorting.

#### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

#### Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



#### `__parallel_sort()` [3/7]

```

template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    multiway_mergesort_exact_tag __parallelism) [inline]

```

Choose multiway mergesort with exact splitting, for parallel sorting.

#### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

#### Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

References [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#), [\\_GLIBCXX\\_CALL](#), and [parallel\\_sort\\_mwms\(\)](#).  
Here is the call graph for this function:



### **\_\_parallel\_sort()** [4/7]

```

template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    multiway_mergesort_sampling_tag __parallelism) [inline]

```

Choose multiway mergesort with splitting by sampling, for parallel sorting.

#### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

#### Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

References [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#), [\\_GLIBCXX\\_CALL](#), and [parallel\\_sort\\_mwms\(\)](#).  
Here is the call graph for this function:



**`__parallel_sort()`** [5/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    multiway_mergesort_tag __parallelism) [inline]
```

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

**Parameters**

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

**Template Parameters**

<code>__stable</code>	Sort stable.
-----------------------	--------------

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::get()`, and `parallel_sort_mwms()`.

Here is the call graph for this function:

**`__parallel_sort()`** [6/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    parallel_tag __parallelism) [inline]
```

Choose a parallel sorting algorithm.

**Parameters**

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `_GLIBCXX_CALL`, `__gnu_parallel::Settings::get()`, and `parallel_sort_mwms()`.

[illegible]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    quicksort_tag __parallelism) [inline]
```

## Parameters

## Template Parameters

Generated by Doxygen

Here is the call graph for this function:



### `__parallel_sort_qs()`

```

template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qs (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __num_threads)

```

Unbalanced quicksort main call.

#### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator input sequence, ignored.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

References `__parallel_sort_qs_conquer()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`, and `__parallel_sort()`.

### `__parallel_sort_qs_conquer()`

```

template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qs_conquer (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __num_threads)

```

Unbalanced quicksort conquer step.

#### Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

References `__parallel_sort_qs_conquer()`, `__parallel_sort_qs_divide()`, and `__gnu_parallel::_Settings::get()`.

Referenced by `__parallel_sort_qs()`, and `__parallel_sort_qs_conquer()`.



**\_\_parallel\_sort\_qs\_divide()**

```
template<typename _RAIter, typename _Compare>
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_sort_qs_divide (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    typename std::iterator_traits< _RAIter >::difference_type __pivot_rank,
    typename std::iterator_traits< _RAIter >::difference_type __num_samples,
    _ThreadIndex __num_threads)
```

Unbalanced quicksort divide step.

**Parameters**

<i>__begin</i>	Begin iterator of subsequence.
<i>__end</i>	End iterator of subsequence.
<i>__comp</i>	Comparator.
<i>__pivot_rank</i>	Desired <i>__rank</i> of the pivot.
<i>__num_samples</i>	Choose pivot from that many samples.
<i>__num_threads</i>	Number of threads that are allowed to work on this part.

References [\\_\\_parallel\\_partition\(\)](#), and [std::min\(\)](#).

Referenced by [\\_\\_parallel\\_sort\\_qs\\_conquer\(\)](#).

**\_\_parallel\_sort\_qsb()**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qsb (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __num_threads)
```

Top-level quicksort routine.

**Parameters**

<i>__begin</i>	Begin iterator of sequence.
<i>__end</i>	End iterator of sequence.
<i>__comp</i>	Comparator.
<i>__num_threads</i>	Number of threads that are allowed to work on this part.

References [\\_\\_qsb\\_conquer\(\)](#), [\\_\\_rd\\_log2\(\)](#), [\\_GLIBCXX\\_CALL](#), and [std::make\\_pair\(\)](#).

Referenced by [\\_\\_parallel\\_sort\(\)](#), and [\\_\\_parallel\\_sort\(\)](#).

**\_\_parallel\_unique\_copy() [1/2]**

```
template<typename _IIter, class _OutputIterator>
_OutputIterator __gnu_parallel::__parallel_unique_copy (
    _IIter __first,
    _IIter __last,
    _OutputIterator __result) [inline]
```

Parallel [std::unique\\_copy\(\)](#), without explicit equality predicate.

## Parameters

<code>__first</code>	Begin iterator of input sequence.
<code>__last</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of result <code>__sequence</code> .

## Returns

End iterator of result `__sequence`.

References [\\_\\_parallel\\_unique\\_copy\(\)](#).

**`__parallel_unique_copy()`** [2/2]

```
template<typename _IIter, class _OutputIterator, class _BinaryPredicate>
_OutputIterator __gnu_parallel::__parallel_unique_copy (
    _IIter __first,
    _IIter __last,
    _OutputIterator __result,
    _BinaryPredicate __binary_pred)
```

Parallel `std::unique_copy()`, w/ `__o` explicit equality predicate.

## Parameters

<code>__first</code>	Begin iterator of input sequence.
<code>__last</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of result <code>__sequence</code> .
<code>__binary_pred</code>	Equality predicate.

## Returns

End iterator of result `__sequence`.

References [\\_\\_equally\\_split\(\)](#), and [\\_GLIBCXX\\_CALL](#).

Referenced by [\\_\\_parallel\\_unique\\_copy\(\)](#).

**`__qsb_conquer()`**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__qsb_conquer (
    _QSBThreadLocal< _RAIter > ** __tls,
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __iam,
    _ThreadIndex __num_threads,
    bool __parent_wait)
```

Quicksort conquer step.

## Parameters

<code>__tls</code>	Array of thread-local storages.
<code>__begin</code>	Begin iterator of subsequence.

## Parameters

<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__iam</code>	Number of the thread processing this function.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

References [\\_\\_qsb\\_conquer\(\)](#), [\\_\\_qsb\\_divide\(\)](#), [\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_elements\\_leftover](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_initial](#), [std::max\(\)](#), and [std::min\(\)](#).

Referenced by [\\_\\_parallel\\_sort\\_qsb\(\)](#), and [\\_\\_qsb\\_conquer\(\)](#).

**`__qsb_divide()`**

```
template<typename _RAIter, typename _Compare>
std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__qsb_divide (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __num_threads)
```

Balanced quicksort divide step.

## Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

## Precondition

`(__end-__begin)>=1`

References [\\_\\_median\\_of\\_three\\_iterators\(\)](#), and [\\_\\_parallel\\_partition\(\)](#).

Referenced by [\\_\\_qsb\\_conquer\(\)](#).

**`__qsb_local_sort_with_helping()`**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__qsb_local_sort_with_helping (
    _QSBThreadLocal<_RAIter> ** __tls,
    _Compare & __comp,
    _ThreadIndex __iam,
    bool __wait)
```

Quicksort step doing load-balanced local sort.

## Parameters

<code>__tls</code>	Array of thread-local storages.
<code>__comp</code>	Comparator.
<code>__iam</code>	Number of the thread processing this function.

References [\\_\\_yield\(\)](#), [\\_GLIBCXX\\_PARALLEL\\_ASSERTIONS](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_elements\\_leftover](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_initial](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_leftover\\_parts](#),

[\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::make\\_pair\(\)](#), and [\\_\\_gnu\\_parallel::\\_Settings::sort\\_qsb\\_base\\_case\\_maximal\\_n](#).  
Referenced by [\\_\\_qsb\\_conquer\(\)](#).

### `__random_number_pow2()`

```
template<typename _RandomNumberGenerator>
int __gnu_parallel::__random_number_pow2 (
    int __logp,
    _RandomNumberGenerator & __rng) [inline]
```

Generate a random number in  $[0, 2^{__logp})$ .

#### Parameters

<code>__logp</code>	Logarithm (basis 2) of the upper range <code>__bound</code> .
<code>__rng</code>	Random number generator to use.

Referenced by [\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#), and [\\_\\_sequential\\_random\\_shuffle\(\)](#).

### `__rd_log2()`

```
template<typename _Size>
_Size __gnu_parallel::__rd_log2 (
    _Size __n) [inline]
```

Calculates the rounded-down logarithm of `__n` for base 2.

#### Parameters

<code>__n</code>	Argument.
------------------	-----------

#### Returns

Returns 0 for any argument  $< 1$ .

Referenced by [\\_\\_gnu\\_parallel::\\_LoserTreeBase<\\_Tp, \\_Compare>::\\_LoserTreeBase\(\)](#), [\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [\\_\\_parallel\\_sort\\_qsb\(\)](#), [\\_\\_round\\_up\\_to\\_pow2\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), [multiseq\\_partition\(\)](#), and [multiseq\\_selection\(\)](#).

### `__round_up_to_pow2()`

```
template<typename _Tp>
_Tp __gnu_parallel::__round_up_to_pow2 (
    _Tp __x)
```

Round up to the next greater power of 2.

#### Parameters

<code>__x</code>	Integer to round up
------------------	---------------------

References [\\_\\_rd\\_log2\(\)](#).

Referenced by [\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), and [multiseq\\_selection\(\)](#).

**\_\_search\_template()**

```
template<typename __RAIter1, typename __RAIter2, typename _Pred>
__RAIter1 __gnu_parallel::__search_template (
    __RAIter1 __begin1,
    __RAIter1 __end1,
    __RAIter2 __begin2,
    __RAIter2 __end2,
    _Pred __pred)
```

Parallel `std::search`.

**Parameters**

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__pred</code>	Find predicate.

**Returns**

Place of finding in first sequences.

References [\\_\\_calc\\_borders\(\)](#), [\\_\\_equally\\_split\(\)](#), [\\_GLIBCXX\\_CALL](#), [std::max\(\)](#), and [std::min\(\)](#).

**\_\_sequential\_multiway\_merge()**

```
template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename ↵
_DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::__sequential_multiway_merge (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
    _DifferenceTp __length,
    _Compare __comp)
```

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

**Parameters**

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.
<code>__sentinel</code>	The sequences have <code>__a</code> <code>__sentinel</code> element.

**Returns**

End iterator of output sequence.

References [\\_\\_is\\_sorted\(\)](#), [\\_\\_merge\\_advance\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_GLIBCXX\\_PARALLEL\\_LENGTH](#), and [std::min\(\)](#).  
Referenced by [multiway\\_merge\(\)](#), [multiway\\_merge\\_sentinels\(\)](#), and [parallel\\_multiway\\_merge\(\)](#).

**`__sequential_random_shuffle()`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__sequential_random_shuffle (
    _RAIter __begin,
    _RAIter __end,
    _RandomNumberGenerator & __rng)
```

Sequential cache-efficient random shuffle.

**Parameters**

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__rng</code>	Random number generator to use.

References [\\_\\_random\\_number\\_pow2\(\)](#), [\\_\\_rd\\_log2\(\)](#), [\\_\\_round\\_up\\_to\\_pow2\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::L2\\_cache\\_size](#), [std::max\(\)](#), [std::min\(\)](#), [std::partial\\_sum\(\)](#), and [\\_\\_gnu\\_parallel::\\_Settings::TLB\\_size](#).

Referenced by [\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#), and [\\_\\_sequential\\_random\\_shuffle\(\)](#).

**`__shrink()`**

```
template<typename _IIter>
void __gnu_parallel::__shrink (
    std::vector<_IIter> & __os_starts,
    size_t & __count_to_two,
    size_t & __range_length)
```

Combines two ranges into one and thus halves the number of ranges.

**Parameters**

<code>__os_starts</code>	Start positions worked on (oversampled).
<code>__count_to_two</code>	Counts up to 2.
<code>__range_length</code>	Current length of a chunk.

References [std::vector<\\_Tp, \\_Alloc>::size\(\)](#).

Referenced by [\\_\\_shrink\\_and\\_double\(\)](#).

**`__shrink_and_double()`**

```
template<typename _IIter>
void __gnu_parallel::__shrink_and_double (
    std::vector<_IIter> & __os_starts,
    size_t & __count_to_two,
    size_t & __range_length,
    const bool __make_twice)
```

Shrinks and doubles the ranges.

**Parameters**

<code>__os_starts</code>	Start positions worked on (oversampled).
<code>__count_to_two</code>	Counts up to 2.
<code>__range_length</code>	Current length of a chunk.

<code>__make_twice</code>	Whether the <code>__os_starts</code> is allowed to be grown or not
---------------------------	--

References [\\_\\_shrink\(\)](#), [std::vector< \\_Tp, \\_Alloc >::resize\(\)](#), and [std::vector< \\_Tp, \\_Alloc >::size\(\)](#).  
Referenced by [list\\_partition\(\)](#).

### **\_\_yield()**

```
void __gnu_parallel::__yield () [inline]
```

Yield control to another thread, without waiting for the end of the time slice.

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#), and [\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

### **list\_partition()**

```
template<typename _IIter, typename _FunctorType>
size_t __gnu_parallel::list_partition (
    const _IIter __begin,
    const _IIter __end,
    _IIter * __starts,
    size_t * __lengths,
    const int __num_parts,
    _FunctorType & __f,
    int __oversampling = 0)
```

Splits a sequence given by input iterators into parts of almost equal size.  
The function needs only one pass over the sequence.

#### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__starts</code>	Start iterators for the resulting parts, dimension <code>__num_parts+1</code> . For convenience, <code>__starts [ __num_parts ]</code> contains the end iterator of the sequence.
<code>__lengths</code>	Length of the resulting parts.
<code>__num_parts</code>	Number of parts to split the sequence into.
<code>__f</code>	Functor to be applied to each element by traversing <code>__it</code>
<code>__oversampling</code>	Oversampling factor. If 0, then the partitions will differ in at most $\sqrt{\text{end} - \text{begin}}$ elements. Otherwise, the ratio between the longest and the shortest part is bounded by $1/(\text{oversampling} \cdot \text{num\_parts})$

#### Returns

Length of the whole sequence.

References [\\_\\_shrink\\_and\\_double\(\)](#), and [std::vector< \\_Tp, \\_Alloc >::size\(\)](#).

### **max()**

```
template<typename _Tp>
const _Tp & __gnu_parallel::max (
    const _Tp & __a,
    const _Tp & __b) [inline]
```

Equivalent to `std::max`.

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#).

**min()**

```
template<typename _Tp>
const _Tp & __gnu_parallel::min (
    const _Tp & __a,
    const _Tp & __b) [inline]
```

Equivalent to `std::min`.

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\\_ed\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\(\)](#), and [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#).

**multiseq\_partition()**

```
template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>
void __gnu_parallel::multiseq_partition (
    _RanSeqs __begin_seqs,
    _RanSeqs __end_seqs,
    _RankType __rank,
    _RankIterator __begin_offsets,
    _Compare __comp = std::less< typename std::iterator_traits<typename _RanSeqs>::value_type:: first_type>::value_type>())
```

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

**Parameters**

<code>__begin_seqs</code>	Begin of the sequence of iterator pairs.
<code>__end_seqs</code>	End of the sequence of iterator pairs.
<code>__rank</code>	The global rank to partition at.
<code>__begin_offsets</code>	A random-access <code>__sequence</code> <code>__begin</code> where the <code>__result</code> will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective <code>__sequence</code> .
<code>__comp</code>	The ordering functor, defaults to <code>std::less&lt;_Tp&gt;</code> .

References [\\_\\_rd\\_log2\(\)](#), [\\_GLIBCXX\\_CALL](#), [std::distance\(\)](#), [std::priority\\_queue<\\_Tp, \\_Sequence, \\_Compare >::empty\(\)](#), [std::make\\_pair\(\)](#), [std::max\(\)](#), [std::min\(\)](#), [std::priority\\_queue<\\_Tp, \\_Sequence, \\_Compare >::pop\(\)](#), [std::priority\\_queue<\\_Tp, \\_Sequence, \\_Compare >::top\(\)](#).

Referenced by [multiway\\_merge\\_exact\\_splitting\(\)](#).

**multiseq\_selection()**

```
template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>
_Tp __gnu_parallel::multiseq_selection (
    _RanSeqs __begin_seqs,
    _RanSeqs __end_seqs,
    _RankType __rank,
    _RankType & __offset,
    _Compare __comp = std::less<_Tp>())
```

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

**Parameters**

<code>__begin_seqs</code>	Begin of the sequence of iterator pairs.
---------------------------	--



## Parameters

<code>__end_seqs</code>	End of the sequence of iterator pairs.
<code>__rank</code>	The global rank to partition at.
<code>__offset</code>	The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
<code>__comp</code>	The ordering functor, defaults to <code>std::less</code> .

References `__rd_log2()`, `__round_up_to_pow2()`, `_GLIBCXX_CALL`, `std::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare>::std::make_pair()`, `std::max()`, `std::min()`, `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::top()`.

**multiway\_merge()**

```
template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
_RAIterOut __gnu_parallel::multiway_merge (
    _RAIterPairIterator __seqs_begin,
    _RAIterPairIterator __seqs_end,
    _RAIterOut __target,
    _DifferenceTp __length,
    _Compare __comp,
    __gnu_parallel::sequential_tag )
```

**Multiway Merge Frontend.**

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

**Example:**

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 10; ++__j)
        sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                    sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

See also

`stable_multiway_merge`

#### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

#### Postcondition

`[__target, return __value)` contains merged `__elements` from the input sequences.

`return __value - __target = min(__length, number of elements in all sequences).`

#### Template Parameters

<code>_RAIterPairIterator</code>	iterator over sequence of pairs of iterators
<code>_RAIterOut</code>	iterator over target sequence
<code>_DifferenceTp</code>	difference type for the sequence
<code>_Compare</code>	strict weak ordering type to compare elements in sequences

#### Parameters

<code>__seqs_begin</code>	<code>__begin</code> of sequence <code>__sequence</code>
<code>__seqs_end</code>	<code>_M_end</code> of sequence <code>__sequence</code>
<code>__target</code>	target sequence to merge to.
<code>__comp</code>	strict weak ordering to use for element comparison.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

#### Returns

`_M_end` iterator of output sequence

References [\\_\\_sequential\\_multiway\\_merge\(\)](#), and [\\_GLIBCXX\\_CALL](#).

#### `multiway_merge_3_variant()`

```
template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename
_RAIter3, typename _DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_3_variant (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    _DifferenceTp __length,
    _Compare __comp)
```

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

## Returns

End iterator of output sequence.

References [\\_GLIBCXX\\_CALL](#).

**multiway\_merge\_4\_variant()**

```
template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename
_RAIter3, typename _DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_4_variant (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    _DifferenceTp __length,
    _Compare __comp)
```

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

## Returns

End iterator of output sequence.

References [\\_GLIBCXX\\_CALL](#).

**multiway\_merge\_exact\_splitting()**

```
template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
void __gnu_parallel::multiway_merge_exact_splitting (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _DifferenceType __length,
```

```

_DifferenceType __total_length,
_Compare __comp,
std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)

```

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

References [\\_\\_equally\\_split\(\)](#), [\\_GLIBCXX\\_PARALLEL\\_LENGTH](#), [std::vector< \\_Tp, \\_Alloc >::begin\(\)](#), [std::vector< \\_Tp, \\_Alloc >::end\(\)](#), [multiseq\\_partition\(\)](#), and [std::vector< \\_Tp, \\_Alloc >::resize\(\)](#).

Referenced by [\\_\\_parallel\\_merge\\_advance\(\)](#).

### **multiway\_merge\_loser\_tree()**

```

template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    _DifferenceTp __length,
    _Compare __comp)

```

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a `LoserTree` class as selected by `_LT`.

Stability is selected through the used `LoserTree` class `_LT`.

At least one non-empty sequence is required.

#### Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

References [\\_GLIBCXX\\_CALL](#), and [\\_GLIBCXX\\_PARALLEL\\_LENGTH](#).

### **multiway\_merge\_loser\_tree\_sentinel()**

```

template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _↵
_DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
    _DifferenceTp __length,
    _Compare __comp)

```

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

## Template Parameters

<code>_UnguardedLoserTree</code>	Loser Tree variant to use for the unguarded merging.
----------------------------------	--

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

## Returns

End iterator of output sequence.

References [\\_\\_is\\_sorted\(\)](#), [\\_GLIBCXX\\_CALL](#), and [multiway\\_merge\\_loser\\_tree\\_unguarded\(\)](#).

## `multiway_merge_loser_tree_unguarded()`

```
template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
    _DifferenceTp __length,
    _Compare __comp)
```

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the `LoserTree` class `_LT`.

Stability is selected by the used `LoserTrees`.

## Precondition

No input will run out of elements during the merge.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

## Returns

End iterator of output sequence.

References [\\_GLIBCXX\\_CALL](#).

Referenced by [multiway\\_merge\\_loser\\_tree\\_sentinel\(\)](#).

**multiway\_merge\_sampling\_splitting()**

```
template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
void __gnu_parallel::multiway_merge_sampling_splitting (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _DifferenceType __length,
    _DifferenceType __total_length,
    _Compare __comp,
    std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)
```

Sampling based splitting for parallel multiway-merge routine.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::merge_oversampling`.

**multiway\_merge\_sentinels()**

```
template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
_RAIterOut __gnu_parallel::multiway_merge_sentinels (
    _RAIterPairIterator __seqs_begin,
    _RAIterPairIterator __seqs_end,
    _RAIterOut __target,
    _DifferenceTp __length,
    _Compare __comp,
    __gnu_parallel::sequential_tag )
```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 11; ++__j)
        sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                     sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

**Precondition**

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

**Postcondition**

`[__target, return __value)` contains merged `__elements` from the input sequences.

`return __value - __target = min(__length, number of elements in all sequences).`

**See also**

`stable_multiway_merge_sentinels`

**Template Parameters**

<code>_RAIterPairIterator</code>	iterator over sequence of pairs of iterators
<code>_RAIterOut</code>	iterator over target sequence
<code>_DifferenceTp</code>	difference type for the sequence
<code>_Compare</code>	strict weak ordering type to compare elements in sequences

**Parameters**

<code>__seqs_begin</code>	<code>__begin</code> of sequence <code>__sequence</code>
<code>__seqs_end</code>	<code>_M_end</code> of sequence <code>__sequence</code>
<code>__target</code>	target sequence to merge to.
<code>__comp</code>	strict weak ordering to use for element comparison.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

**Returns**

`_M_end` iterator of output sequence

References [\\_\\_sequential\\_multiway\\_merge\(\)](#), and [\\_GLIBCXX\\_CALL](#).

**parallel\_multiway\_merge()**

```
template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename ↵
_DifferenceTp, typename _Splitter, typename _Compare>
_RAIter3 __gnu_parallel::parallel_multiway_merge (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    _Splitter __splitter,
    _DifferenceTp __length,
    _Compare __comp,
    _ThreadIndex __num_threads)
```

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

Must not be called if the number of sequences is 1.

## Template Parameters

<code>_Splitter</code>	functor to split input (either <code>__exact</code> or sampling based)
<code>__stable</code>	Stable merging incurs a performance penalty.
<code>__sentinel</code>	Ignored.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

## Returns

End iterator of output sequence.

References [\\_\\_is\\_sorted\(\)](#), [\\_\\_sequential\\_multiway\\_merge\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_GLIBCXX\\_PARALLEL\\_LENGTH](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::make\\_pair\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::merge\\_oversampling](#), and [std::min\(\)](#).

Referenced by [\\_\\_parallel\\_merge\\_advance\(\)](#).

**`parallel_sort_mwms()`**

```
template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void __gnu_parallel::parallel_sort_mwms (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __num_threads)
```

PMWMS main call.

## Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads to use.

References [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_pieces](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_samples](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_source](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_starts](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_temporary](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [parallel\\_sort\\_mwms\\_pu\(\)](#), and [\\_\\_gnu\\_parallel::\\_Settings::sort\\_mwms\\_oversampling](#).

Referenced by [\\_\\_parallel\\_sort\(\)](#), [\\_\\_parallel\\_sort\(\)](#), [\\_\\_parallel\\_sort\(\)](#), and [\\_\\_parallel\\_sort\(\)](#).

**`parallel_sort_mwms_pu()`**

```
template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void __gnu_parallel::parallel_sort_mwms_pu (
    _PMWMSortingData<_RAIter> * __sd,
    _Compare & __comp)
```

PMWMS code executed by each thread.



## Parameters

<code>__sd</code>	Pointer to algorithm data.
<code>__comp</code>	Comparator.

References [\\_\\_gnu\\_parallel::\\_PMWSSortingData<\\_RAIter>::M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::\\_PMWSSortingData<\\_RAIter>::M\\_source](#), [\\_\\_gnu\\_parallel::\\_PMWSSortingData<\\_RAIter>::M\\_starts](#), [\\_\\_gnu\\_parallel::\\_PMWSSortingData<\\_RAIter>::M\\_temporary](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::make\\_pair\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::sort\\_mwms\\_oversampling](#), and [std::uninitialized\\_copy\(\)](#).

Referenced by [parallel\\_sort\\_mwms\(\)](#).

#### 4.6.5 Variable Documentation

##### `_CASable_bits`

```
const int __gnu_parallel::_CASable_bits [static]
```

Number of bits of `_CASable`.

Referenced by [\\_\\_decode2\(\)](#), and [\\_\\_encode2\(\)](#).

##### `_CASable_mask`

```
const _CASable __gnu_parallel::_CASable_mask [static]
```

`_CASable` with the right half of bits set to 1.

Referenced by [\\_\\_decode2\(\)](#).

### 4.7 `__gnu_pbds` Namespace Reference

#### Classes

- struct [associative\\_tag](#)
- class [basic\\_branch](#)
- struct [basic\\_branch\\_tag](#)
- class [basic\\_hash\\_table](#)
- struct [basic\\_hash\\_tag](#)
- struct [basic\\_invalidation\\_guarantee](#)
- struct [binary\\_heap\\_tag](#)
- struct [binomial\\_heap\\_tag](#)
- class [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)
- class [cc\\_hash\\_table](#)
- struct [cc\\_hash\\_tag](#)
- struct [container\\_error](#)
- struct [container\\_tag](#)
- struct [container\\_traits](#)
- struct [container\\_traits\\_base](#)
- struct [container\\_traits\\_base< binary\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< binomial\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< cc\\_hash\\_tag >](#)
- struct [container\\_traits\\_base< gp\\_hash\\_tag >](#)
- struct [container\\_traits\\_base< list\\_update\\_tag >](#)
- struct [container\\_traits\\_base< ov\\_tree\\_tag >](#)
- struct [container\\_traits\\_base< pairing\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< pat\\_trie\\_tag >](#)
- struct [container\\_traits\\_base< rb\\_tree\\_tag >](#)

- struct `container_traits_base< rc_binomial_heap_tag >`
- struct `container_traits_base< splay_tree_tag >`
- struct `container_traits_base< thin_heap_tag >`
- class `direct_mask_range_hashing`
- class `direct_mod_range_hashing`
- class `gp_hash_table`
- struct `gp_hash_tag`
- class `hash_exponential_size_policy`
- class `hash_load_check_resize_trigger`
- class `hash_prime_size_policy`
- class `hash_standard_resize_policy`
- struct `insert_error`
- struct `join_error`
- class `linear_probe_fn`
- class `list_update`
- struct `list_update_tag`
- class `lu_counter_policy`
- class `lu_move_to_front_policy`
- struct `null_node_update`
- struct `null_type`
- struct `ov_tree_tag`
- struct `pairing_heap_tag`
- struct `pat_trie_tag`
- struct `point_invalidation_guarantee`
- class `priority_queue`
- struct `priority_queue_tag`
- class `quadratic_probe_fn`
- struct `range_invalidation_guarantee`
- struct `rb_tree_tag`
- struct `rc_binomial_heap_tag`
- struct `resize_error`
- class `sample_probe_fn`
- class `sample_range_hashing`
- class `sample_ranged_hash_fn`
- class `sample_ranged_probe_fn`
- class `sample_resize_policy`
- class `sample_resize_trigger`
- class `sample_size_policy`
- class `sample_tree_node_update`
- struct `sample_trie_access_traits`
- class `sample_trie_node_update`
- struct `sample_update_policy`
- struct `sequence_tag`
- struct `splay_tree_tag`
- struct `string_tag`
- struct `thin_heap_tag`
- class `tree`
- class `tree_order_statistics_node_update`
- struct `tree_tag`
- class `trie`
- class `trie_order_statistics_node_update`

- class [trie\\_prefix\\_search\\_node\\_update](#)
- struct [trie\\_string\\_access\\_traits](#)
- struct [trie\\_tag](#)
- struct [trivial\\_iterator\\_tag](#)

## Typedefs

- typedef void [trivial\\_iterator\\_difference\\_type](#)

## Functions

- void [\\_\\_throw\\_container\\_error](#) ()
- void [\\_\\_throw\\_insert\\_error](#) ()
- void [\\_\\_throw\\_join\\_error](#) ()
- void [\\_\\_throw\\_resize\\_error](#) ()

## Variables

- `template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename ↵  
_Alloc>  
detail::integral_constant< int, Reverse > trie\_string\_access\_traits< String, Min_E_Val, Max_E_Val, Reverse,  
_Alloc >::s_rev_ind`

### 4.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

## 4.8 [\\_\\_gnu\\_sequential](#) Namespace Reference

### 4.8.1 Detailed Description

GNU sequential classes for public use.

## 4.9 [abi](#) Namespace Reference

### 4.9.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`.

A brief overview of an ABI is given in the libstdc++ FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5\\_8](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8)).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

## 4.10 [std](#) Namespace Reference

### Namespaces

- namespace [\\_\\_debug](#)
- namespace [\\_\\_detail](#)
- namespace [\\_\\_parallel](#)

- namespace [chrono](#)
- namespace [decimal](#)
- namespace [experimental](#)
- namespace [filesystem](#)
- namespace [literals](#)
- namespace [placeholders](#)
- namespace [regex\\_constants](#)
- namespace [rel\\_ops](#)
- namespace [this\\_thread](#)
- namespace [tr1](#)
- namespace [tr2](#)

## Classes

- class [\\_\\_basic\\_future](#)
- class [\\_\\_codecvt\\_abstract\\_base](#)
- class [\\_\\_ctype\\_abstract\\_base](#)
- struct [\\_\\_is\\_fast\\_hash](#)
- struct [\\_\\_is\\_location\\_invariant](#)
- class [\\_\\_new\\_allocator](#)
- struct [\\_\\_numeric\\_limits\\_base](#)
- struct [\\_Base\\_bitset](#)
- struct [\\_Base\\_bitset< 0 >](#)
- struct [\\_Base\\_bitset< 1 >](#)
- class [\\_Bind](#)
- class [\\_Bind\\_result](#)
- class [\\_Deque\\_base](#)
- struct [\\_Deque\\_iterator](#)
- class [\\_Function\\_base](#)
- struct [\\_Fwd\\_list\\_base](#)
- struct [\\_Fwd\\_list\\_const\\_iterator](#)
- struct [\\_Fwd\\_list\\_iterator](#)
- struct [\\_Fwd\\_list\\_node](#)
- struct [\\_Fwd\\_list\\_node\\_base](#)
- class [\\_List\\_base](#)
- struct [\\_List\\_const\\_iterator](#)
- struct [\\_List\\_iterator](#)
- struct [\\_List\\_node](#)
- class [\\_Not\\_fn](#)
- struct [\\_Placeholder](#)
- struct [\\_Sp\\_ebo\\_helper< \\_Nm, \\_Tp, false >](#)
- struct [\\_Sp\\_ebo\\_helper< \\_Nm, \\_Tp, true >](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Vector\\_base](#)
- struct [add\\_const](#)
- struct [add\\_cv](#)
- struct [add\\_lvalue\\_reference](#)
- struct [add\\_pointer](#)
- struct [add\\_rvalue\\_reference](#)
- struct [add\\_volatile](#)
- struct [adopt\\_lock\\_t](#)

- struct [aligned\\_storage](#)
- struct [aligned\\_union](#)
- struct [alignment\\_of](#)
- class [allocator](#)
- struct [allocator\\_traits](#)
- struct [allocator\\_traits< allocator< \\_Tp > >](#)
- struct [allocator\\_traits< allocator< void > >](#)
- struct [allocator\\_traits< pmr::polymorphic\\_allocator< \\_Tp > >](#)
- struct [array](#)
- struct [atomic](#)
- struct [atomic< \\_Tp \\* >](#)
- struct [atomic\\_flag](#)
- class [auto\\_ptr](#)
- struct [auto\\_ptr\\_ref](#)
- class [back\\_insert\\_iterator](#)
- class [bad\\_alloc](#)
- class [bad\\_cast](#)
- class [bad\\_exception](#)
- class [bad\\_function\\_call](#)
- class [bad\\_typeid](#)
- class [bad\\_weak\\_ptr](#)
- class [basic\\_filebuf](#)
- class [basic\\_fstream](#)
- class [basic\\_ifstream](#)
- class [basic\\_ios](#)
- class [basic\\_iostream](#)
- class [basic\\_istream](#)
- class [basic\\_istreambuf\\_iterator](#)
- class [basic\\_ofstream](#)
- class [basic\\_ostream](#)
- class [basic\\_ostreambuf\\_iterator](#)
- class [basic\\_regex](#)
- class [basic\\_streambuf](#)
- class [basic\\_string](#)
- class [basic\\_string\\_view](#)
- class [basic\\_stringbuf](#)
- class [basic\\_stringstream](#)
- class [bernoulli\\_distribution](#)
- struct [bidirectional\\_iterator\\_tag](#)
- struct [binary\\_function](#)
- class [binary\\_negate](#)
- class [binder1st](#)
- class [binder2nd](#)
- class [binomial\\_distribution](#)
- class [bitset](#)
- class [cauchy\\_distribution](#)
- struct [char\\_traits](#)
- struct [char\\_traits< \\_\\_gnu\\_cxx::character< \\_Value, \\_Int, \\_St > >](#)
- struct [char\\_traits< wchar\\_t >](#)
- class [chi\\_squared\\_distribution](#)
- class [codecvt](#)

- class `codecvt< _InternT, _ExternT, encoding_state >`
- class `codecvt< char, char, mbstate_t >`
- class `codecvt< char16_t, char, mbstate_t >`
- class `codecvt< char32_t, char, mbstate_t >`
- class `codecvt< wchar_t, char, mbstate_t >`
- class `codecvt_base`
- class `codecvt_byname`
- class `common_iterator`
- struct `common_type`
- struct `common_type< chrono::duration< _Rep, _Period > >`
- struct `common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >`
- struct `common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >`
- struct `common_type< chrono::time_point< _Clock, _Duration > >`
- struct `common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >`
- struct `common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >`
- struct `compare_three_way_result`
- class `complex`
- class `complex< double >`
- class `complex< float >`
- class `complex< long double >`
- class `condition_variable`
- class `condition_variable_any`
- struct `conditional`
- class `const_mem_fun1_ref_t`
- class `const_mem_fun1_t`
- class `const_mem_fun_ref_t`
- class `const_mem_fun_t`
- struct `contiguous_iterator_tag`
- class `copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- class `counted_iterator`
- class `ctype`
- class `ctype< char >`
- class `ctype< wchar_t >`
- struct `ctype_base`
- class `ctype_byname`
- class `ctype_byname< char >`
- struct `decay`
- struct `default_delete`
- struct `default_delete< _Tp[]>`
- struct `default_sentinel_t`
- struct `defer_lock_t`
- class `deque`
- struct `destroying_delete_t`
- class `discard_block_engine`
- class `discrete_distribution`
- struct `divides`
- struct `divides< void >`
- class `domain_error`
- struct `enable_if`
- class `enable_shared_from_this`
- class `encoding_state`

- struct `equal_to`
- struct `equal_to< void >`
- class `error_category`
- class `error_code`
- class `error_condition`
- class `exception`
- class `exception_ptr`
- class `exponential_distribution`
- struct `extent`
- class `extreme_value_distribution`
- class `fisher_f_distribution`
- struct `forward_iterator_tag`
- class `forward_list`
- class `fpos`
- struct `from_chars_result`
- class `front_insert_iterator`
- class `function< _Res(_ArgTypes...)>`
- class `function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- class `future`
- class `future< _Res & >`
- class `future< void >`
- class `future_error`
- class `gamma_distribution`
- class `geometric_distribution`
- struct `greater`
- struct `greater< void >`
- struct `greater_equal`
- struct `greater_equal< void >`
- class `gslice`
- class `gslice_array`
- struct `has_virtual_destructor`
- struct `hash`
- struct `hash< __debug::bitset< _Nb > >`
- struct `hash< __debug::vector< bool, _Alloc > >`
- struct `hash< __gnu_cxx::__u16vstring >`
- struct `hash< __gnu_cxx::__u32vstring >`
- struct `hash< __gnu_cxx::__vstring >`
- struct `hash< __gnu_cxx::__wvstring >`
- struct `hash< __gnu_cxx::throw_value_limit >`
- struct `hash< __gnu_cxx::throw_value_random >`
- struct `hash< __gnu_debug::basic_string< _CharT > >`
- struct `hash< __shared_ptr< _Tp, _Lp > >`
- struct `hash< _Tp * >`
- struct `hash< basic_string< char, char_traits< char >, _Alloc > >`
- struct `hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >`
- struct `hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >`
- struct `hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >`
- struct `hash< bool >`
- struct `hash< char >`
- struct `hash< char16_t >`
- struct `hash< char32_t >`

- struct `hash< double >`
- struct `hash< error_code >`
- struct `hash< error_condition >`
- struct `hash< experimental::optional< _Tp > >`
- struct `hash< experimental::shared_ptr< _Tp > >`
- struct `hash< float >`
- struct `hash< int >`
- struct `hash< long >`
- struct `hash< long double >`
- struct `hash< long long >`
- struct `hash< shared_ptr< _Tp > >`
- struct `hash< short >`
- struct `hash< signed char >`
- struct `hash< thread::id >`
- struct `hash< type_index >`
- struct `hash< unique_ptr< _Tp, _Dp > >`
- struct `hash< unsigned char >`
- struct `hash< unsigned int >`
- struct `hash< unsigned long >`
- struct `hash< unsigned long long >`
- struct `hash< unsigned short >`
- struct `hash< wchar_t >`
- struct `hash<::bitset< _Nb > >`
- struct `hash<::vector< bool, _Alloc > >`
- struct `identity`
- class `independent_bits_engine`
- class `indirect_array`
- class `initializer_list`
- struct `input_iterator_tag`
- class `insert_iterator`
- struct `integer_sequence`
- struct `integral_constant`
- class `invalid_argument`
- class `ios_base`
- struct `is_abstract`
- struct `is_arithmetic`
- struct `is_array`
- struct `is_assignable`
- struct `is_base_of`
- struct `is_bind_expression`
- struct `is_bind_expression< _Bind< _Signature > >`
- struct `is_bind_expression< _Bind_result< _Result, _Signature > >`
- struct `is_bind_expression< const _Bind< _Signature > >`
- struct `is_bind_expression< const _Bind_result< _Result, _Signature > >`
- struct `is_bind_expression< const volatile _Bind< _Signature > >`
- struct `is_bind_expression< const volatile _Bind_result< _Result, _Signature > >`
- struct `is_bind_expression< volatile _Bind< _Signature > >`
- struct `is_bind_expression< volatile _Bind_result< _Result, _Signature > >`
- struct `is_class`
- struct `is_compound`
- struct `is_const`



- struct [is\\_constructible](#)
- struct [is\\_copy\\_assignable](#)
- struct [is\\_copy\\_constructible](#)
- struct [is\\_default\\_constructible](#)
- struct [is\\_destructible](#)
- struct [is\\_empty](#)
- struct [is\\_enum](#)
- struct [is\\_error\\_code\\_enum](#)
- struct [is\\_error\\_code\\_enum](#)< future\_errc >
- struct [is\\_error\\_condition\\_enum](#)
- struct [is\\_floating\\_point](#)
- struct [is\\_function](#)
- struct [is\\_fundamental](#)
- struct [is\\_integral](#)
- struct [is\\_layout\\_compatible](#)
- struct [is\\_literal\\_type](#)
- struct [is\\_lvalue\\_reference](#)
- struct [is\\_member\\_function\\_pointer](#)
- struct [is\\_member\\_object\\_pointer](#)
- struct [is\\_member\\_pointer](#)
- struct [is\\_move\\_assignable](#)
- struct [is\\_move\\_constructible](#)
- struct [is\\_nothrow\\_assignable](#)
- struct [is\\_nothrow\\_constructible](#)
- struct [is\\_nothrow\\_copy\\_assignable](#)
- struct [is\\_nothrow\\_copy\\_constructible](#)
- struct [is\\_nothrow\\_default\\_constructible](#)
- struct [is\\_nothrow\\_destructible](#)
- struct [is\\_nothrow\\_move\\_assignable](#)
- struct [is\\_nothrow\\_move\\_constructible](#)
- struct [is\\_object](#)
- struct [is\\_placeholder](#)
- struct [is\\_placeholder](#)< \_Placeholder< \_Num > >
- struct [is\\_pod](#)
- struct [is\\_pointer](#)
- struct [is\\_pointer\\_interconvertible\\_base\\_of](#)
- struct [is\\_polymorphic](#)
- struct [is\\_reference](#)
- struct [is\\_rvalue\\_reference](#)
- struct [is\\_same](#)
- struct [is\\_scalar](#)
- struct [is\\_signed](#)
- struct [is\\_standard\\_layout](#)
- struct [is\\_trivial](#)
- struct [is\\_trivially\\_assignable](#)
- struct [is\\_trivially\\_constructible](#)
- struct [is\\_trivially\\_copy\\_assignable](#)
- struct [is\\_trivially\\_copy\\_constructible](#)
- struct [is\\_trivially\\_copyable](#)
- struct [is\\_trivially\\_default\\_constructible](#)
- struct [is\\_trivially\\_destructible](#)

- struct [is\\_trivially\\_move\\_assignable](#)
- struct [is\\_trivially\\_move\\_constructible](#)
- struct [is\\_union](#)
- struct [is\\_unsigned](#)
- struct [is\\_void](#)
- struct [is\\_volatile](#)
- class [istream\\_iterator](#)
- class [istreambuf\\_iterator](#)
- struct [iterator](#)
- struct [iterator\\_traits](#)
- struct [iterator\\_traits< \\_Tp \\* >](#)
- class [length\\_error](#)
- struct [less](#)
- struct [less< void >](#)
- struct [less\\_equal](#)
- struct [less\\_equal< void >](#)
- class [linear\\_congruential\\_engine](#)
- class [list](#)
- class [locale](#)
- class [lock\\_guard](#)
- class [logic\\_error](#)
- struct [logical\\_and](#)
- struct [logical\\_and< void >](#)
- struct [logical\\_not](#)
- struct [logical\\_not< void >](#)
- struct [logical\\_or](#)
- struct [logical\\_or< void >](#)
- class [lognormal\\_distribution](#)
- struct [make\\_signed](#)
- struct [make\\_unsigned](#)
- class [map](#)
- class [mask\\_array](#)
- class [match\\_results](#)
- class [mem\\_fun1\\_ref\\_t](#)
- class [mem\\_fun1\\_t](#)
- class [mem\\_fun\\_ref\\_t](#)
- class [mem\\_fun\\_t](#)
- class [mersenne\\_twister\\_engine](#)
- class [messages](#)
- struct [messages\\_base](#)
- class [messages\\_byname](#)
- struct [minus](#)
- struct [minus< void >](#)
- struct [modulus](#)
- struct [modulus< void >](#)
- class [money\\_base](#)
- class [money\\_get](#)
- class [money\\_put](#)
- class [moneypunct](#)
- class [moneypunct\\_byname](#)
- class [move\\_iterator](#)

- class `move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- class `move_sentinel`
- class `multimap`
- struct `multiplies`
- struct `multiplies< void >`
- class `multiset`
- class `mutex`
- struct `negate`
- struct `negate< void >`
- class `negative_binomial_distribution`
- class `nested_exception`
- class `normal_distribution`
- struct `not_equal_to`
- struct `not_equal_to< void >`
- class `num_get`
- class `num_put`
- struct `numeric_limits`
- struct `numeric_limits< bool >`
- struct `numeric_limits< char >`
- struct `numeric_limits< char16_t >`
- struct `numeric_limits< char32_t >`
- struct `numeric_limits< double >`
- struct `numeric_limits< float >`
- struct `numeric_limits< int >`
- struct `numeric_limits< long >`
- struct `numeric_limits< long double >`
- struct `numeric_limits< long long >`
- struct `numeric_limits< short >`
- struct `numeric_limits< signed char >`
- struct `numeric_limits< unsigned char >`
- struct `numeric_limits< unsigned int >`
- struct `numeric_limits< unsigned long >`
- struct `numeric_limits< unsigned long long >`
- struct `numeric_limits< unsigned short >`
- struct `numeric_limits< wchar_t >`
- class `num_punct`
- class `num_punct_byname`
- struct `once_flag`
- class `ostream_iterator`
- class `ostreambuf_iterator`
- class `out_of_range`
- struct `output_iterator_tag`
- class `overflow_error`
- struct `owner_less`
- struct `owner_less< shared_ptr< _Tp > >`
- struct `owner_less< void >`
- struct `owner_less< weak_ptr< _Tp > >`
- class `packaged_task< _Res(_ArgTypes...)>`
- struct `pair`
- class `piecewise_constant_distribution`
- struct `piecewise_construct_t`

- class [piecewise\\_linear\\_distribution](#)
- struct [plus](#)
- class [pointer\\_to\\_binary\\_function](#)
- class [pointer\\_to\\_unary\\_function](#)
- struct [pointer\\_traits](#)
- struct [pointer\\_traits< \\_Tp \\* >](#)
- class [poisson\\_distribution](#)
- class [priority\\_queue](#)
- class [promise](#)
- class [promise< \\_Res & >](#)
- class [promise< void >](#)
- class [queue](#)
- struct [random\\_access\\_iterator\\_tag](#)
- class [random\\_device](#)
- class [range\\_error](#)
- struct [rank](#)
- struct [ratio](#)
- struct [ratio\\_equal](#)
- struct [ratio\\_greater](#)
- struct [ratio\\_greater\\_equal](#)
- struct [ratio\\_less](#)
- struct [ratio\\_less\\_equal](#)
- struct [ratio\\_not\\_equal](#)
- class [raw\\_storage\\_iterator](#)
- class [recursive\\_mutex](#)
- class [recursive\\_timed\\_mutex](#)
- class [reference\\_wrapper](#)
- class [regex\\_error](#)
- class [regex\\_iterator](#)
- class [regex\\_token\\_iterator](#)
- class [regex\\_traits](#)
- struct [remove\\_all\\_extents](#)
- struct [remove\\_const](#)
- struct [remove\\_cv](#)
- struct [remove\\_extent](#)
- struct [remove\\_pointer](#)
- struct [remove\\_reference](#)
- struct [remove\\_volatile](#)
- struct [result\\_of](#)
- class [reverse\\_iterator](#)
- class [runtime\\_error](#)
- class [scoped\\_allocator\\_adaptor](#)
- class [seed\\_seq](#)
- class [set](#)
- class [shared\\_future](#)
- class [shared\\_future< \\_Res & >](#)
- class [shared\\_future< void >](#)
- class [shared\\_lock](#)
- class [shared\\_ptr](#)
- class [shared\\_timed\\_mutex](#)
- class [shuffle\\_order\\_engine](#)

- class [slice](#)
- class [slice\\_array](#)
- class [stack](#)
- class [student\\_t\\_distribution](#)
- class [sub\\_match](#)
- class [subtract\\_with\\_carry\\_engine](#)
- class [system\\_error](#)
- class [thread](#)
- class [time\\_base](#)
- class [time\\_get](#)
- class [time\\_get\\_byname](#)
- class [time\\_put](#)
- class [time\\_put\\_byname](#)
- class [timed\\_mutex](#)
- struct [to\\_chars\\_result](#)
- struct [try\\_to\\_lock\\_t](#)
- class [tuple](#)
- class [tuple<\\_T1, \\_T2>](#)
- struct [tuple\\_element](#)
- struct [tuple\\_element<0, pair<\\_Tp1, \\_Tp2>>](#)
- struct [tuple\\_element<1, pair<\\_Tp1, \\_Tp2>>](#)
- struct [tuple\\_element<\\_\\_i, tuple<\\_Types...>>](#)
- struct [tuple\\_element<\\_Ind, array<\\_Tp, \\_Nm>>](#)
- struct [tuple\\_size](#)
- struct [tuple\\_size<array<\\_Tp, \\_Nm>>](#)
- struct [tuple\\_size<pair<\\_Tp1, \\_Tp2>>](#)
- struct [tuple\\_size<tuple<\\_Elements...>>](#)
- struct [type\\_index](#)
- class [type\\_info](#)
- struct [unary\\_function](#)
- class [unary\\_negate](#)
- class [underflow\\_error](#)
- struct [underlying\\_type](#)
- class [uniform\\_int\\_distribution](#)
- class [uniform\\_real\\_distribution](#)
- class [unique\\_lock](#)
- class [unique\\_ptr](#)
- class [unique\\_ptr<\\_Tp\[\], \\_Dp>](#)
- class [unordered\\_map](#)
- class [unordered\\_multimap](#)
- class [unordered\\_multiset](#)
- class [unordered\\_set](#)
- struct [uses\\_allocator](#)
- struct [uses\\_allocator<tuple<\\_Types...>, \\_Alloc>](#)
- class [valarray](#)
- class [vector](#)
- class [vector<bool, \\_Alloc>](#)
- class [wbuffer\\_convert](#)
- class [weak\\_ptr](#)
- class [weibull\\_distribution](#)
- class [wstring\\_convert](#)

## Typedefs

- `template<typename _Tp>`  
using [\\_\\_allocator\\_base](#)
- `typedef unsigned char` **\_\_atomic\_flag\_data\_type**
- `typedef FILE` **\_\_c\_file**
- `typedef __locale_t` **\_\_c\_locale**
- `typedef __gthread_mutex_t` **\_\_c\_lock**
- `template<bool _Cond, typename _If, typename _Else>`  
using **\_\_conditional\_t**
- `template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up>::value>::type, size_t = tuple_size<_Tp>::value>`  
using **\_\_enable\_if\_has\_tuple\_size**
- `template<bool _Cond, typename _Tp = void>`  
using **\_\_enable\_if\_t**
- `template<typename _Fn, typename _Op>`  
using **\_\_function\_guide\_t**
- `template<typename _Func, typename _SfinaeType>`  
using **\_\_has\_is\_transparent\_t**
- `template<typename _ToElementType, typename _FromElementType>`  
using **\_\_is\_array\_convertible**
- `template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>`  
using **\_\_is\_socketlike**
- `template<typename _Iter>`  
using **\_\_iter\_category\_t**
- `template<typename _InputIterator>`  
using **\_\_iter\_key\_t**
- `template<typename _InputIterator>`  
using **\_\_iter\_to\_alloc\_t**
- `template<typename _InputIterator>`  
using **\_\_iter\_val\_t**
- `template<typename _Tp, typename _Up>`  
using **\_\_like\_t**
- `template<typename _Ptr, typename _Tp>`  
using [\\_\\_ptr\\_rebind](#)
- `template<typename _Is, typename _Tp>`  
using **\_\_rvalue\_stream\_extraction\_t**
- `template<typename _Os, typename _Tp>`  
using **\_\_rvalue\_stream\_insertion\_t**
- `typedef basic\_string< char >` **\_\_sso\_string**
- `template<size_t __i, typename _Tp>`  
using **\_\_tuple\_element\_t**
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>`  
using **\_\_umap\_hashtable**
- `template<bool _Cache>`  
using [\\_\\_umap\\_traits](#)
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>`  
using **\_\_ummap\_hashtable**
- `template<bool _Cache>`  
using [\\_\\_ummap\\_traits](#)

- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>`  
`using __umset_hashtable`
- `template<bool _Cache>`  
`using \_\_umset\_traits`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>`  
`using __uset_hashtable`
- `template<bool _Cache>`  
`using \_\_uset\_traits`
- `template<typename _Fn, typename... _Args>`  
`using _Bind_back_t`
- `template<typename _Fn, typename... _Args>`  
`using _Bind_front_t`
- `typedef unsigned long _Bit_type`
- `template<typename _Path, typename _Result = _Path, typename _Path2 = decltype(std::declval<_Path&>().make_preferred().filename())>`  
`using _If_fs_path`
- `template<typename _Iter>`  
`using _RequireInputIter`
- `template<typename _Tp>`  
`using add\_lvalue\_reference\_t`
- `template<typename _Tp>`  
`using add\_pointer\_t`
- `template<typename _Tp>`  
`using add\_rvalue\_reference\_t`
- `template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>`  
`using aligned\_storage\_t`
- `template<size_t _Len, typename... _Types>`  
`using aligned_union_t`
- `typedef atomic< bool > atomic\_bool`
- `typedef atomic< char > atomic\_char`
- `typedef atomic< char16_t > atomic\_char16\_t`
- `typedef atomic< char32_t > atomic\_char32\_t`
- `typedef atomic< int > atomic\_int`
- `typedef atomic< int16_t > atomic\_int16\_t`
- `typedef atomic< int32_t > atomic\_int32\_t`
- `typedef atomic< int64_t > atomic\_int64\_t`
- `typedef atomic< int8_t > atomic\_int8\_t`
- `typedef atomic< int_fast16_t > atomic\_int\_fast16\_t`
- `typedef atomic< int_fast32_t > atomic\_int\_fast32\_t`
- `typedef atomic< int_fast64_t > atomic\_int\_fast64\_t`
- `typedef atomic< int_fast8_t > atomic\_int\_fast8\_t`
- `typedef atomic< int_least16_t > atomic\_int\_least16\_t`
- `typedef atomic< int_least32_t > atomic\_int\_least32\_t`
- `typedef atomic< int_least64_t > atomic\_int\_least64\_t`
- `typedef atomic< int_least8_t > atomic\_int\_least8\_t`
- `typedef atomic< intmax_t > atomic\_intmax\_t`
- `typedef atomic< intptr_t > atomic\_intptr\_t`
- `typedef atomic< long long > atomic\_llong`
- `typedef atomic< long > atomic\_long`
- `typedef atomic< ptrdiff_t > atomic\_ptrdiff\_t`
- `typedef atomic< signed char > atomic\_schar`

- typedef [atomic](#)< short > [atomic\\_short](#)
- typedef [atomic](#)< size\_t > [atomic\\_size\\_t](#)
- typedef [atomic](#)< unsigned char > [atomic\\_uchar](#)
- typedef [atomic](#)< unsigned int > [atomic\\_uint](#)
- typedef [atomic](#)< uint16\_t > [atomic\\_uint16\\_t](#)
- typedef [atomic](#)< uint32\_t > [atomic\\_uint32\\_t](#)
- typedef [atomic](#)< uint64\_t > [atomic\\_uint64\\_t](#)
- typedef [atomic](#)< uint8\_t > [atomic\\_uint8\\_t](#)
- typedef [atomic](#)< uint\_fast16\_t > [atomic\\_uint\\_fast16\\_t](#)
- typedef [atomic](#)< uint\_fast32\_t > [atomic\\_uint\\_fast32\\_t](#)
- typedef [atomic](#)< uint\_fast64\_t > [atomic\\_uint\\_fast64\\_t](#)
- typedef [atomic](#)< uint\_fast8\_t > [atomic\\_uint\\_fast8\\_t](#)
- typedef [atomic](#)< uint\_least16\_t > [atomic\\_uint\\_least16\\_t](#)
- typedef [atomic](#)< uint\_least32\_t > [atomic\\_uint\\_least32\\_t](#)
- typedef [atomic](#)< uint\_least64\_t > [atomic\\_uint\\_least64\\_t](#)
- typedef [atomic](#)< uint\_least8\_t > [atomic\\_uint\\_least8\\_t](#)
- typedef [atomic](#)< uintmax\_t > [atomic\\_uintmax\\_t](#)
- typedef [atomic](#)< uintptr\_t > [atomic\\_uintptr\\_t](#)
- typedef [atomic](#)< unsigned long long > [atomic\\_ullong](#)
- typedef [atomic](#)< unsigned long > [atomic\\_ulong](#)
- typedef [atomic](#)< unsigned short > [atomic\\_ushort](#)
- typedef [atomic](#)< wchar\_t > [atomic\\_wchar\\_t](#)
- using **atto**
- using **centi**
- typedef [match\\_results](#)< const char \* > **cmatch**
- template<typename... \_Ts>  
using **common\_comparison\_category\_t**
- template<typename... \_Tp>  
using [common\\_type\\_t](#)
- template<typename \_Tp, typename \_Up = \_Tp>  
using [compare\\_three\\_way\\_result\\_t](#)
- template<bool \_Cond, typename \_Iftrue, typename \_Iffalse>  
using [conditional\\_t](#)
- typedef [regex\\_iterator](#)< const char \* > **cregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const char \* > [cregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const char \* > [csub\\_match](#)
- using **deca**
- template<typename \_Tp>  
using [decay\\_t](#)
- using **deci**
- typedef [minstd\\_rand0](#) **default\_random\_engine**
- template<bool \_Cond, typename \_Tp = void>  
using [enable\\_if\\_t](#)
- using **exa**
- using [false\\_type](#)
- using **femto**
- typedef [basic\\_filebuf](#)< char > [filebuf](#)
- typedef [basic\\_fstream](#)< char > [fstream](#)
- using **giga**
- using **hecto**
- typedef [basic\\_ifstream](#)< char > [ifstream](#)



---

- `template<size_t... _Idx>`  
  using `index_sequence`
- `template<typename... _Types>`  
  using `index_sequence_for`
- `typedef basic_ios< char > ios`
- `typedef basic_iostream< char > iostream`
- `typedef basic_istream< char > istream`
- `typedef basic_istreamstream< char > istringstream`
- using **kilo**
- `typedef shuffle_order_engine< minstd_rand0, 256 > knuth_b`
- `template<size_t _Num>`  
  using `make_index_sequence`
- `template<typename _Tp, _Tp _Num>`  
  using `make_integer_sequence`
- `template<typename _Tp>`  
  using `make_signed_t`
- `template<typename _Tp>`  
  using `make_unsigned_t`
- using **mega**
- using **micro**
- using **milli**
- `typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > minstd_rand`
- `typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > minstd_rand0`
- `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > mt19937`
- `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > mt19937_64`
- using **nano**
- `typedef void(* new_handler) ()`
- `typedef decltype(nullptr) nullptr_t`
- `typedef basic_ofstream< char > ofstream`
- `typedef basic_ostream< char > ostream`
- `typedef basic_ostringstream< char > ostringstream`
- using **peta**
- using **pico**
- `typedef __PTRDIFF_TYPE__ ptrdiff_t`
- `typedef discard_block_engine< ranlux24_base, 223, 23 > ranlux24`
- `typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > ranlux24_base`
- `typedef discard_block_engine< ranlux48_base, 389, 11 > ranlux48`
- `typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > ranlux48_base`
- `template<typename _R1, typename _R2>`  
  using `ratio_add`
- `template<typename _R1, typename _R2>`  
  using `ratio_divide`
- `template<typename _R1, typename _R2>`  
  using `ratio_multiply`
- `template<typename _R1, typename _R2>`  
  using `ratio_subtract`
- `typedef basic_regex< char > regex`
- `template<typename _Tp>`  
  using `remove_all_extents_t`

---

- template<typename \_Tp>  
using [remove\\_extent\\_t](#)
- template<typename \_Tp>  
using [remove\\_pointer\\_t](#)
- template<typename \_Tp>  
using [remove\\_reference\\_t](#)
- template<typename \_Tp>  
using [result\\_of\\_t](#)
- typedef [\\_\\_SIZE\\_TYPE\\_\\_](#) **size\_t**
- typedef [match\\_results](#)< string::const\_iterator > **smatch**
- typedef [regex\\_iterator](#)< string::const\_iterator > **sregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< string::const\_iterator > **sregex\_token\_iterator**
- typedef [sub\\_match](#)< string::const\_iterator > **ssub\_match**
- typedef [basic\\_streambuf](#)< char > **streambuf**
- typedef long long **streamoff**
- typedef [fpos](#)< mbstate\_t > **streampos**
- typedef ptrdiff\_t **streamsize**
- typedef [basic\\_string](#)< char > **string**
- using **string\_view**
- typedef [basic\\_stringbuf](#)< char > **stringbuf**
- typedef [basic\\_stringstream](#)< char > **stringstream**
- using **tera**
- typedef void(\* [terminate\\_handler](#)) ()
- using [true\\_type](#)
- template<size\_t \_\_i, typename \_Tp>  
using **tuple\_element\_t**
- typedef [fpos](#)< mbstate\_t > **u16streampos**
- typedef [basic\\_string](#)< char16\_t > **u16string**
- using **u16string\_view**
- typedef [fpos](#)< mbstate\_t > **u32streampos**
- typedef [basic\\_string](#)< char32\_t > **u32string**
- using **u32string\_view**
- template<typename \_Tp>  
using [underlying\\_type\\_t](#)
- typedef void(\* [unexpected\\_handler](#)) ()
- typedef [match\\_results](#)< const wchar\_t \* > **wcmatch**
- typedef [regex\\_iterator](#)< const wchar\_t \* > **wcregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const wchar\_t \* > **wcregex\_token\_iterator**
- typedef [sub\\_match](#)< const wchar\_t \* > **wcsub\_match**
- typedef [basic\\_filebuf](#)< wchar\_t > **wfilebuf**
- typedef [basic\\_fstream](#)< wchar\_t > **wfstream**
- typedef [basic\\_ifstream](#)< wchar\_t > **wifstream**
- typedef [basic\\_ios](#)< wchar\_t > **wios**
- typedef [basic\\_iostream](#)< wchar\_t > **wiostream**
- typedef [basic\\_istream](#)< wchar\_t > **wistream**
- typedef [basic\\_istreamstream](#)< wchar\_t > **wistreamstream**
- typedef [basic\\_ofstream](#)< wchar\_t > **wofstream**
- typedef [basic\\_ostream](#)< wchar\_t > **wostream**
- typedef [basic\\_ostreamstream](#)< wchar\_t > **wostreamstream**
- typedef [basic\\_regex](#)< wchar\_t > **wregex**
- typedef [match\\_results](#)< wstring::const\_iterator > **wsmatch**

- typedef [regex\\_iterator](#)< wstring::const\_iterator > **wsregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< wstring::const\_iterator > **wsregex\_token\_iterator**
- typedef [sub\\_match](#)< wstring::const\_iterator > **wssub\_match**
- typedef [basic\\_streambuf](#)< wchar\_t > **wstreambuf**
- typedef [fpos](#)< mbstate\_t > **wstreampos**
- typedef [basic\\_string](#)< wchar\_t > **wstring**
- using **wstring\_view**
- typedef [basic\\_stringbuf](#)< wchar\_t > **wstringbuf**
- typedef [basic\\_stringstream](#)< wchar\_t > **wstringstream**

## Enumerations

- enum { **\_S\_chunk\_size** }
- enum { **\_S\_word\_bit** }
- enum **\_ios\_Fmtflags** {  
**\_S\_boolalpha**, **\_S\_dec**, **\_S\_fixed**, **\_S\_hex**,  
**\_S\_internal**, **\_S\_left**, **\_S\_oct**, **\_S\_right**,  
**\_S\_scientific**, **\_S\_showbase**, **\_S\_showpoint**, **\_S\_showpos**,  
**\_S\_skipws**, **\_S\_unitbuf**, **\_S\_uppercase**, **\_S\_adjustfield**,  
**\_S\_basefield**, **\_S\_floatfield**, **\_S\_ios\_fmtflags\_end**, **\_S\_ios\_fmtflags\_max**,  
**\_S\_ios\_fmtflags\_min** }
- enum **\_ios\_iostate** {  
**\_S\_goodbit**, **\_S\_badbit**, **\_S\_eofbit**, **\_S\_failbit**,  
**\_S\_ios\_iostate\_end**, **\_S\_ios\_iostate\_max**, **\_S\_ios\_iostate\_min** }
- enum **\_ios\_Openmode** {  
**\_S\_app**, **\_S\_ate**, **\_S\_bin**, **\_S\_in**,  
**\_S\_out**, **\_S\_trunc**, **\_S\_noreplace**, **\_S\_ios\_openmode\_end**,  
**\_S\_ios\_openmode\_max**, **\_S\_ios\_openmode\_min** }
- enum **\_ios\_Seekdir** { **\_S\_beg**, **\_S\_cur**, **\_S\_end**, **\_S\_ios\_seekdir\_end** }
- enum **\_Lock\_policy**
- enum **\_Manager\_operation** { **\_\_get\_type\_info**, **\_\_get\_func\_ptr**, **\_\_clone\_func**, **\_\_destroy\_func** }
- enum **\_Rb\_tree\_color** { **\_S\_red**, **\_S\_black** }
- enum class **align\_val\_t** : size\_t
- enum class **chars\_format** { **scientific**, **fixed**, **hex**, **general** }
- enum **codecvt\_mode** { **consume\_header**, **generate\_header**, **little\_endian** }
- enum class **cv\_status** { **no\_timeout**, **timeout** }
- enum class **errc** {  
**address\_family\_not\_supported**, **address\_in\_use**, **address\_not\_available**, **already\_connected**,  
**argument\_list\_too\_long**, **argument\_out\_of\_domain**, **bad\_address**, **bad\_file\_descriptor**,  
**broken\_pipe**, **connection\_aborted**, **connection\_already\_in\_progress**, **connection\_refused**,  
**connection\_reset**, **cross\_device\_link**, **destination\_address\_required**, **device\_or\_resource\_busy**,  
**directory\_not\_empty**, **executable\_format\_error**, **file\_exists**, **file\_too\_large**,  
**filename\_too\_long**, **function\_not\_supported**, **host\_unreachable**, **illegal\_byte\_sequence**,  
**inappropriate\_io\_control\_operation**, **interrupted**, **invalid\_argument**, **invalid\_seek**,  
**io\_error**, **is\_a\_directory**, **message\_size**, **network\_down**,  
**network\_reset**, **network\_unreachable**, **no\_buffer\_space**, **no\_child\_process**,  
**no\_lock\_available**, **no\_message**, **no\_protocol\_option**, **no\_space\_on\_device**,  
**no\_such\_device\_or\_address**, **no\_such\_device**, **no\_such\_file\_or\_directory**, **no\_such\_process**,  
**not\_a\_directory**, **not\_a\_socket**, **not\_connected**, **not\_enough\_memory**,  
**operation\_in\_progress**, **operation\_not\_permitted**, **operation\_not\_supported**, **operation\_would\_block**,  
**permission\_denied**, **protocol\_not\_supported**, **read\_only\_file\_system**, **resource\_deadlock\_would\_occur**,  
**resource\_unavailable\_try\_again**, **result\_out\_of\_range**, **timed\_out**, **too\_many\_files\_open\_in\_system**,  
**too\_many\_files\_open**, **too\_many\_links**, **too\_many\_symbolic\_link\_levels**, **wrong\_protocol\_type** }

- enum `float_denorm_style` { `denorm_indeterminate` , `denorm_absent` , `denorm_present` }
- enum `float_round_style` { `round_indeterminate` , `round_toward_zero` , `round_to_nearest` , `round_toward_infinity` , `round_toward_neg_infinity` }
- enum class `future_errc` { `future_already_retrieved` , `promise_already_satisfied` , `no_state` , `broken_promise` }
- enum class `future_status` { `ready` , `timeout` , `deferred` }
- enum class `io_errc` { `stream` }
- enum class `launch` { `async` , `deferred` }
- enum class `memory_order` : int { `relaxed` , `consume` , `acquire` , `release` , `acq_rel` , `seq_cst` }
- enum class `pointer_safety` { `relaxed` , `preferred` , `strict` }

## Functions

- template<typename \_CharT>  
\_CharT \* **\_\_add\_grouping** (\_CharT \*\_\_s, \_CharT \_\_sep, const char \*\_\_gbeg, size\_t \_\_gsize, const \_CharT \*\_\_first, const \_CharT \*\_\_last)
- template<typename \_Tp>  
constexpr \_Tp \* **\_\_addressof** (\_Tp &\_\_r) noexcept
- template<typename \_ForwardIterator, typename \_BinaryPredicate>  
constexpr \_ForwardIterator **\_\_adjacent\_find** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_BinaryPredicate \_\_binary\_pred)
- template<typename \_RandomAccessIterator, typename \_Distance, typename \_Tp, typename \_Compare>  
constexpr void **\_\_adjust\_heap** (\_RandomAccessIterator \_\_first, \_Distance \_\_holeIndex, \_Distance \_\_len, \_Tp \_\_value, \_Compare \_\_comp)
- template<typename \_BidirectionalIterator, typename \_Distance>  
constexpr void **\_\_advance** (\_BidirectionalIterator &\_\_i, \_Distance \_\_n, `bidirectional_iterator_tag`)
- template<typename \_InputIterator, typename \_Distance>  
constexpr void **\_\_advance** (\_InputIterator &\_\_i, \_Distance \_\_n, `input_iterator_tag`)
- template<typename \_OutputIterator, typename \_Distance>  
void **\_\_advance** (\_OutputIterator &, \_Distance, `output_iterator_tag`)=delete
- template<typename \_RandomAccessIterator, typename \_Distance>  
constexpr void **\_\_advance** (\_RandomAccessIterator &\_\_i, \_Distance \_\_n, `random_access_iterator_tag`)
- template<typename \_Tp, \_Lock\_policy \_Lp = \_\_default\_lock\_policy, typename \_Alloc, typename... \_Args>  
\_\_shared\_ptr< \_Tp, \_Lp > **\_\_allocate\_shared** (const \_Alloc &\_\_a, \_Args &&... \_\_args)
- template<bool \_IsMove, typename \_OutIter, typename \_InIter>  
constexpr void **\_\_assign\_one** (\_OutIter &\_\_out, \_InIter &\_\_in)
- template<typename \_Facet>  
const \_Facet & **\_\_check\_facet** (const \_Facet \*\_\_f)
- template<typename \_RandomAccessIterator, typename \_Distance, typename \_Compare>  
constexpr void **\_\_chunk\_insertion\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Distance \_\_chunk\_size, \_Compare \_\_comp)
- template<typename \_Tp>  
\_Tp **\_\_complex\_abs** (const `complex`< \_Tp > &\_\_z)
- \_\_complex\_\_ double **\_\_complex\_acos** (\_\_complex\_\_ double \_\_z)
- \_\_complex\_\_ float **\_\_complex\_acos** (\_\_complex\_\_ float \_\_z)
- \_\_complex\_\_ long double **\_\_complex\_acos** (const \_\_complex\_\_ long double &\_\_z)
- template<typename \_Tp>  
`std::complex`< \_Tp > **\_\_complex\_acos** (const `std::complex`< \_Tp > &\_\_z)
- \_\_complex\_\_ double **\_\_complex\_acosh** (\_\_complex\_\_ double \_\_z)

- `__complex__ float __complex_acosh (__complex__ float __z)`
- `__complex__ long double __complex_acosh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`_Tp __complex_arg (const complex< _Tp > &__z)`
- `__complex__ double __complex_asin (__complex__ double __z)`
- `__complex__ float __complex_asin (__complex__ float __z)`
- `__complex__ long double __complex_asin (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_asin (const std::complex< _Tp > &__z)`
- `__complex__ double __complex_asinh (__complex__ double __z)`
- `__complex__ float __complex_asinh (__complex__ float __z)`
- `__complex__ long double __complex_asinh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_asinh (const std::complex< _Tp > &__z)`
- `__complex__ double __complex_atan (__complex__ double __z)`
- `__complex__ float __complex_atan (__complex__ float __z)`
- `__complex__ long double __complex_atan (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_atan (const std::complex< _Tp > &__z)`
- `__complex__ double __complex_atanh (__complex__ double __z)`
- `__complex__ float __complex_atanh (__complex__ float __z)`
- `__complex__ long double __complex_atanh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `complex< double > __complex_proj (const complex< double > &__z)`
- `complex< float > __complex_proj (const complex< float > &__z)`
- `complex< long double > __complex_proj (const complex< long double > &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_tan (const complex< _Tp > &__z)`

- `template<typename _Tp>`  
`complex< _Tp > __complex_tanh (const complex< _Tp > &__z)`
- `int __convert_from_v (const __c_locale &__cloc, char *__out, const int __size, const char *__fmt,...)`
- `template<typename _Tp>`  
`void __convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<> void __convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<> void __convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<> void __convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI __copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr __gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > __copy_move_a (_II, _II, const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>`  
`constexpr ::__gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > __copy_move_a (const ::__gnu_debug::Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI __copy_move_a (const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > __copy_move_a1 (::Deque_iterator< _ITp, _IRef, _IPtr > __first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>`  
`_OI __copy_move_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Tp>`  
`__gnu_cxx::enable_if< __is_random_access_iter< _II >::value, ::Deque_iterator< _Tp, _Tp &, _Tp * > >::type __copy_move_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI __copy_move_a1 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::type __copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > > >)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT > >::type __copy_move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _InIter, typename _Sent, typename _OutIter>`  
`constexpr _OutIter __copy_move_a2 (_InIter __first, _Sent __last, _OutIter __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::type __copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > > >)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT > >::type __copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, _CharT * >::type __copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ::Deque_iterator< _CharT, _CharT &, _CharT * > >::type __copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT > > __first,`

```

istreambuf_iterator<_CharT, char_traits<_CharT>> __last, ::_Deque_iterator<_CharT, _CharT &, _CharT *
> __result)
• template<bool _IsMove, typename _CharT>
    __gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT *>::__type __copy_move_a2(istreambuf_iterator<
    _CharT, char_traits<_CharT>>, istreambuf_iterator<_CharT, char_traits<_CharT>>, _CharT *)
• template<bool _IsMove, typename _II, typename _OI>
    constexpr _OI __copy_move_backward_a(_II __first, _II __last, _OI __result)
• template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
    constexpr __gnu_debug::Safe_iterator<_Ite, _Seq, _Cat> __copy_move_backward_a(_II __first, _II __last, const
    ::__gnu_debug::Safe_iterator<_Ite, _Seq, _Cat> &)
• template<bool _IsMove, typename _Ite, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
    constexpr ::__gnu_debug::Safe_iterator<_OIte, _OSeq, _OCat> __copy_move_backward_a(const
    ::__gnu_debug::Safe_iterator<_Ite, _ISeq, _ICat> &, const ::__gnu_debug::Safe_iterator<_Ite, _ISeq,
    _ICat> &, const ::__gnu_debug::Safe_iterator<_OIte, _OSeq, _OCat> &)
• template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
    constexpr _OI __copy_move_backward_a(const ::__gnu_debug::Safe_iterator<_Ite, _Seq, _Cat> &, const
    ::__gnu_debug::Safe_iterator<_Ite, _Seq, _Cat> &, _OI)
• template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
    ::_Deque_iterator<_OTp, _OTp &, _OTp *> __copy_move_backward_a1(::_Deque_iterator<_ITp, _IRef,
    _IPtr> __first, ::_Deque_iterator<_ITp, _IRef, _IPtr> __last, ::_Deque_iterator<_OTp, _OTp &, _OTp *>
    __result)
• template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
    _OI __copy_move_backward_a1(::_Deque_iterator<_Tp, _Ref, _Ptr> __first, ::_Deque_iterator<_Tp, _Ref,
    _Ptr> __last, _OI __result)
• template<bool _IsMove, typename _BI1, typename _BI2>
    constexpr _BI2 __copy_move_backward_a1(_BI1 __first, _BI1 __last, _BI2 __result)
• template<bool _IsMove, typename _II, typename _Tp>
    __gnu_cxx::__enable_if<__is_random_access_iter<_II>::__value, ::_Deque_iterator<_Tp, _Tp &, _Tp *>
    >::__type __copy_move_backward_a1(_II __first, _II __last, ::_Deque_iterator<_Tp, _Tp &, _Tp *> __
    result)
• template<bool _IsMove, typename _BI1, typename _BI2>
    constexpr _BI2 __copy_move_backward_a2(_BI1 __first, _BI1 __last, _BI2 __result)
• template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
    _OI __copy_move_backward_dit(::_Deque_iterator<_Tp, _Ref, _Ptr> __first, ::_Deque_iterator<_Tp, _Ref,
    _Ptr> __last, _OI __result)
• template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
    _OI __copy_move_dit(::_Deque_iterator<_Tp, _Ref, _Ptr> __first, ::_Deque_iterator<_Tp, _Ref, _Ptr>
    __last, _OI __result)
• template<typename _InputIterator, typename _Size, typename _OutputIterator>
    constexpr _OutputIterator __copy_n_a(_InputIterator __first, _Size __n, _OutputIterator __result, bool)
• template<typename _CharT, typename _Size>
    __gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT *>::__type __copy_n_a(istreambuf_iterator<
    _CharT> __it, _Size __n, _CharT * __result, bool __strict)
• template<typename _CharT, typename _Size>
    __gnu_cxx::__enable_if<__is_char<_CharT>::__value, ::_Deque_iterator<_CharT, _CharT &, _CharT *>
    >::__type __copy_n_a(istreambuf_iterator<_CharT, char_traits<_CharT>> __it, _Size __size, ::_Deque_
    iterator<_CharT, _CharT &, _CharT *> __result, bool __strict)
• template<typename _CharT, typename _Size>
    __gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT *>::__type __copy_n_a(istreambuf_iterator<
    _CharT, char_traits<_CharT>>, _Size, _CharT *, bool)
• template<typename _CharT, typename _Traits>
    streamsize __copy_streambufs(basic_streambuf<_CharT, _Traits> * __sbin, basic_streambuf<_CharT, _
    Traits> * __sbout)

```



- `template<typename _CharT, typename _Traits>`  
`streamsize __copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sout, bool &__ineof)`
- `template<> streamsize __copy_streambufs_eof (basic_streambuf< char > *__sbin, basic_streambuf< char > *__sout, bool &__ineof)`
- `template<> streamsize __copy_streambufs_eof (basic_streambuf< wchar_t > *__sbin, basic_streambuf< wchar_t > *__sout, bool &__ineof)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr iterator_traits< _InputIterator >::difference_type __count_if ( _InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `constexpr size_t __deque_buf_size (size_t __size)`
- `template<typename _Tp>`  
`ptrdiff_t __distance (:: _List_const_iterator< _Tp >, :: _List_const_iterator< _Tp >, input_iterator_tag)`
- `template<typename _Tp>`  
`ptrdiff_t __distance (:: _List_iterator< _Tp >, :: _List_iterator< _Tp >, input_iterator_tag)`
- `template<bool _Const, typename _Ptr>`  
`ptrdiff_t __distance ( _list:: _Iterator< _Const, _Ptr > __first, _list:: _Iterator< _Const, _Ptr > __last, input_iterator_tag __tag)`
- `template<typename _InputIterator>`  
`constexpr iterator_traits< _InputIterator >::difference_type __distance ( _InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _OutputIterator>`  
`void __distance ( _OutputIterator, _OutputIterator, output_iterator_tag)=delete`
- `template<typename _RandomAccessIterator>`  
`constexpr iterator_traits< _RandomAccessIterator >::difference_type __distance ( _RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn>`  
`bool __do_str_codecvt (const _InChar * __first, const _InChar * __last, _OutStr &__outstr, const _Codecvt &__cvt, _State &__state, size_t &__count, _Fn __fn)`
- `template<typename _I1, typename _I2>`  
`constexpr bool __equal4 ( _I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _BinaryPredicate>`  
`constexpr bool __equal4 ( _I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2>`  
`constexpr bool __equal_aux ( _I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _I1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool __equal_aux ( _I1, _I1, const :: _gnu_debug:: Safe_iterator< _I2, _Seq2, _Cat2 > &)`
- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool __equal_aux (const :: _gnu_debug:: Safe_iterator< _I1, _Seq1, _Cat1 > &, const :: _gnu_debug:: Safe_iterator< _I1, _Seq1, _Cat1 > &, _I2)`
- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool __equal_aux (const :: _gnu_debug:: Safe_iterator< _I1, _Seq1, _Cat1 > &, const :: _gnu_debug:: Safe_iterator< _I1, _Seq1, _Cat1 > &, const :: _gnu_debug:: Safe_iterator< _I2, _Seq2, _Cat2 > &)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II>`  
`__gnu_cxx:: __enable_if< __is_random_access_iter< _II >:: __value, bool >:: __type __equal_aux1 (:: Deque_iterator< _Tp, _Ref, _Ptr > __first1, :: Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool __equal_aux1 (:: Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, :: Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, :: Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)`
- `template<typename _II, typename _Tp, typename _Ref, typename _Ptr>`  
`__gnu_cxx:: __enable_if< __is_random_access_iter< _II >:: __value, bool >:: __type __equal_aux1 ( _II __first1, _II __last1, :: Deque_iterator< _Tp, _Ref, _Ptr > __first2)`



- `template<typename _I1, typename _I2>`  
`constexpr bool __equal_aux1 (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II>`  
`bool __equal_dit (const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__first1, const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__last1, _II __first2)`
- `template<typename _ForwardIterator, typename _Tp, typename _CompareItTp, typename _CompareTplt>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > __equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _CompareItTp __comp_it_val, _CompareTplt __comp_val_it)`
- `template<typename _Tp, typename _Up = _Tp>`  
`constexpr _Tp __exchange (_Tp &__obj, _Up &&__new_val)`
- `template<typename _Flte, typename _Tp>`  
`constexpr void __fill_a (_Flte __first, _Flte __last, const _Tp &__value)`
- `template<typename _Lte, typename _Seq, typename _Cat, typename _Tp>`  
`constexpr void __fill_a (const ::__gnu_debug::__Safe_iterator< _Lte, _Seq, _Cat > &, const ::__gnu_debug::__Safe_iterator< _Lte, _Seq, _Cat > &, const _Tp &)`
- `template<typename _Lte, typename _Cont, typename _Tp>`  
`constexpr void __fill_a1 (::__gnu_cxx::__normal_iterator< _Lte, _Cont > __first, ::__gnu_cxx::__normal_iterator< _Lte, _Cont > __last, const _Tp &__value)`
- `constexpr void __fill_a1 (::_Bit_iterator, ::_Bit_iterator, const bool &)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void __fill_a1 (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Up, typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_byte< _Up >::__value &&(__are_same< _Up, _Tp >::__value||__memcpable_integer< _Tp >::__width), void >::__type __fill_a1 (_Up *__first, _Up *__last, const _Tp &__x)`
- `template<typename _Tp, typename _VTP>`  
`void __fill_a1 (const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTP &__value)`
- `constexpr void __fill_bvector (_Bit_type *__v, unsigned int __first, unsigned int __last, bool __x) noexcept`
- `constexpr void __fill_bvector_n (_Bit_type *, size_t, bool) noexcept`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator __fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator __fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::output_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator __fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::random_access_iterator_tag)`
- `template<typename _Lte, typename _Seq, typename _Cat, typename _Size, typename _Tp>`  
`constexpr ::__gnu_debug::__Safe_iterator< _Lte, _Seq, _Cat > __fill_n_a (const ::__gnu_debug::__Safe_iterator< _Lte, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator __fill_n_a1 (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate>`  
`constexpr _BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 __find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _Iterator, typename _Predicate>`  
`constexpr _Iterator __find_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Predicate, typename _Distance>`  
`constexpr _InputIterator \_\_find\_if\_not\_n (_InputIterator __first, _Distance &__len, _Predicate __pred)`
- `template<typename _Tp, typename... _Types>`  
`constexpr size_t \_\_find\_uniq\_type\_in\_pack ()`
- `from_chars_result \_\_from\_chars\_bfloat16\_t (const char * __first, const char * __last, float & __value, chars\_format __fmt=chars_format::general) noexcept`
- `from_chars_result \_\_from\_chars\_float16\_t (const char * __first, const char * __last, float & __value, chars\_format __fmt=chars_format::general) noexcept`
- `template<typename _EuclideanRingElement>`  
`constexpr _EuclideanRingElement \_\_gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _IntType, typename _UniformRandomBitGenerator>`  
`pair< _IntType, _IntType > \_\_gen\_two\_uniform\_ints (_IntType __b0, _IntType __b1, _UniformRandomBitGenerator && __g)`
- `template<size_t __i, typename _Head, typename... _Tail>`  
`constexpr _Head & \_\_get\_helper (_Tuple_impl< __i, _Head, _Tail... > & __t) noexcept`
- `template<size_t __i, typename _Head, typename... _Tail>`  
`constexpr const _Head & \_\_get\_helper (const _Tuple_impl< __i, _Head, _Tail... > & __t) noexcept`
- `template<size_t __i, typename... _Types>`  
`\_\_enable\_if\_t<(__i >= sizeof...( _Types))> \_\_get\_helper (const tuple< _Types... > &) = delete`
- `void \_\_glibcxx\_assert\_fail (const char * __file, int __line, const char * __function, const char * __condition) noexcept`
- `template<typename _Tp>`  
`size_t \_\_iconv\_adapter (size_t (* __func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char ** __inbuf, size_t * __inbytes, char ** __outbuf, size_t * __outbytes)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare>`  
`constexpr bool \_\_includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`void \_\_inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`void \_\_inplace\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _CharT, typename _ValueT>`  
`int \_\_int\_to\_char (_CharT * __bufend, _ValueT __v, const _CharT * __lit, ios\_base::fmtflags __flags, bool __dec)`
- `template<typename _Tp, typename _Up = typename __inv_unwrap< _Tp >::type>`  
`constexpr _Up && \_\_invfwd (typename remove\_reference< _Tp >::type & __t) noexcept`
- `template<typename _Callable, typename... _Args>`  
`constexpr __invoke_result< _Callable, _Args... >::type \_\_invoke (_Callable && __fn, _Args &&... __args) noexcept(\_\_is\_nothrow\_invocable< _Callable, _Args... >::value)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res \_\_invoke\_impl (__invoke_memfun_deref, _MemFun && __f, _Tp && __t, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res \_\_invoke\_impl (__invoke_memfun_ref, _MemFun && __f, _Tp && __t, _Args &&... __args)`
- `template<typename _Res, typename _MemPtr, typename _Tp>`  
`constexpr _Res \_\_invoke\_impl (__invoke_memobj_deref, _MemPtr && __f, _Tp && __t)`
- `template<typename _Res, typename _MemPtr, typename _Tp>`  
`constexpr _Res \_\_invoke\_impl (__invoke_memobj_ref, _MemPtr && __f, _Tp && __t)`
- `template<typename _Res, typename _Fn, typename... _Args>`  
`constexpr _Res \_\_invoke\_impl (__invoke_other, _Fn && __f, _Args &&... __args)`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`constexpr enable\_if\_t< is\_invocable\_r\_v< _Res, _Callable, _Args... >, _Res > \_\_invoke\_r (_Callable && __fn, _Args &&... __args) noexcept(is\_nothrow\_invocable\_r\_v< _Res, _Callable, _Args... >)`
- `constexpr bool \_\_is\_constant\_evaluated () noexcept`

- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance>`  
`constexpr bool __is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance>`  
`constexpr bool __is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator>`  
`constexpr bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare>`  
`constexpr _Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare &__comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool __is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool __is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator __is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare &__comp)`
- `template<typename _CharT, typename _Traits>`  
`void __istream_extract (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)`
- `void __istream_extract (istream &, char *, streamsize)`
- `template<typename _Iter>`  
`constexpr iterator_traits< _Iter >::iterator_category __iterator_category (const _Iter &)`
- `template<typename _Tp1, typename _Ref, typename _Ptr, typename _Tp2>`  
`int __lex_cmp_dit (::Deque_iterator< _Tp1, _Ref, _Ptr > __first1, ::Deque_iterator< _Tp1, _Ref, _Ptr > __last1, const _Tp2 * __first2, const _Tp2 * __last2)`
- `template<typename _I1, typename _I2>`  
`constexpr bool __lexicographical_compare_aux (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _Ite2, typename _Seq2, typename _Cat2>`  
`constexpr bool __lexicographical_compare_aux (_I1 __first1, _I1 __last1, const ::__gnu_debug::__Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::__Safe_iterator< _Ite2, _Seq2, _Cat2 > &__last2)`
- `template<typename _I1, typename _Iter2, typename _Seq2, typename _Cat2>`  
`constexpr bool __lexicographical_compare_aux (_I1, _I1, const ::__gnu_debug::__Safe_iterator< _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::__Safe_iterator< _Iter2, _Seq2, _Cat2 > &)`
- `template<typename _Ite1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool __lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, _Cat1 > &__first1, const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, _I2 __first2, _I2 __last2)`
- `template<typename _Ite1, typename _Seq1, typename _Cat1, typename _Ite2, typename _Seq2, typename _Cat2>`  
`constexpr bool __lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, _Cat1 > &__first1, const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, const ::__gnu_debug::__Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::__Safe_iterator< _Ite2, _Seq2, _Cat2 > &__last2)`
- `template<typename _Iter1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool __lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator< _Iter1, _Seq1, _Cat1 > &, _I2, _I2)`
- `template<typename _Iter1, typename _Seq1, typename _Cat1, typename _Iter2, typename _Seq2, typename _Cat2>`  
`constexpr bool __lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator< _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::__Safe_iterator< _Iter2, _Seq2, _Cat2 > &)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool __lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)`

- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>`  
`bool __lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 * __first2, _Tp2 * __last2)`
- `template<typename _II1, typename _II2>`  
`constexpr bool __lexicographical_compare_aux1 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool __lexicographical_compare_aux1 (_Tp1 * __first1, _Tp1 * __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)`
- `template<typename _II1, typename _II2, typename _Compare>`  
`constexpr bool __lexicographical_compare_impl (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, ↵ Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp __lg (_Tp __n)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator __lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `template<typename... _BoundArgs, typename... _Args>`  
`constexpr auto __make_bound_args (_Args &&... __args)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void __make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare & ↵ __comp)`
- `template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits< ↵ Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>>`  
`constexpr _ReturnType __make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move ↵ iterator<_Tp*>>>>`  
`constexpr _ReturnType __make_move_if_noexcept_iterator (_Tp * __i)`
- `template<typename _Iterator>`  
`constexpr reverse_iterator< _Iterator > __make_reverse_iterator (_Iterator __i)`
- `template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename... _Args>`  
`__shared_ptr< _Tp, _Lp > __make_shared (_Args &&... __args)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator __max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare ↵ comp)`
- `template<typename _Tp, typename _Up>`  
`constexpr int __memcmp (const _Tp * __first1, const _Up * __first2, size_t __num)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵ _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>`  
`void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator ↵ __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>`  
`void __merge_adaptive_resize (_BidirectionalIterator __first, _BidirectionalIterator __middle, _Bidirectional ↵ Iterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare ↵ __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare>`  
`void __merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _Random ↵ AccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>`  
`void __merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer ↵ buffer, _Compare __comp)`

- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare>`  
`void __merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr auto __min_cmp (_Tp __x, _Tp __y)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator __min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr pair<_ForwardIterator, _ForwardIterator> __minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair<_InputIterator1, _InputIterator2> __mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair<_InputIterator1, _InputIterator2> __mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _Iterator>`  
`constexpr _Iterator __miter_base (_Iterator __it)`
- `template<typename _Iterator, typename _Compare>`  
`constexpr void __move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`void __move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare>`  
`void __move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool __next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Ite, typename _Seq>`  
`constexpr decltype(std::__niter_base(std::declval<_Ite>())) __niter_base (const ::__gnu_debug::__Safe_iterator<_Ite, _Seq, std::random_access_iterator_tag> &__it) noexcept(std::is_nothrow_copy_constructible<_Ite>::value)`
- `void __once_proxy (void)`
- `template<typename _BidirectionalIterator, typename _Predicate>`  
`constexpr _BidirectionalIterator __partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare &__comp)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool __prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>`  
`constexpr void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare &__comp)`

- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>`  
`_Out_iter __regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex<_Ch_type, _Rx_traits> &__e, const _Ch_type * __fmt, size_t __len, regex_constants::match_flag_type __flags)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator __remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator __remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>`  
`constexpr _OutputIterator __replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp & __new_value)`
- `template<typename _BidirectionalIterator>`  
`constexpr void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator>`  
`constexpr void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator>`  
`constexpr _BidirectionalIterator __rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator __rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)`
- `template<typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator __rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance>`  
`_BidirectionalIterator1 __rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBitGenerator>`  
`_OutputIterator __sample (_ForwardIterator __first, _ForwardIterator __last, forward_iterator_tag, _OutputIterator __out, _Cat, _Size __n, _UniformRandomBitGenerator && __g)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator>`  
`_RandomAccessIterator __sample (_InputIterator __first, _InputIterator __last, input_iterator_tag, _RandomAccessIterator __out, random_access_iterator_tag, _Size __n, _UniformRandomBitGenerator && __g)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 __search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator __search_n_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate>`  
`constexpr _RandomAccessIter __search_n_aux (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, _UnaryPredicate __unary_pred, std::random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`



- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, ↵`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp>`  
`_Tp * __shared_ptr_deref (_Tp * __p)`
- `constexpr long long __size_to_integer (double __n)`
- `constexpr long long __size_to_integer (float __n)`
- `constexpr int __size_to_integer (int __n)`
- `constexpr long __size_to_integer (long __n)`
- `constexpr long long __size_to_integer (long double __n)`
- `constexpr long long __size_to_integer (long long __n)`
- `constexpr unsigned __size_to_integer (unsigned __n)`
- `constexpr unsigned long __size_to_integer (unsigned long __n)`
- `constexpr unsigned long long __size_to_integer (unsigned long long __n)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void __sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__↵`  
`comp)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`_ForwardIterator __stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>`  
`_ForwardIterator __stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __↵`  
`pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`void __stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>`  
`void __stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random↵`  
`AccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare>`  
`void __stable_sort_adaptive_resize (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer`  
`__buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool __str_codecvt_in (const char * __first, const char * __last, basic\_string< _CharT, _Traits, _Alloc > &__↵`  
`outstr, const codecvt< _CharT, char, _State > & __cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool __str_codecvt_in (const char * __first, const char * __last, basic\_string< _CharT, _Traits, _Alloc > &__↵`  
`outstr, const codecvt< _CharT, char, _State > & __cvt, _State & __state, size_t & __count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool __str_codecvt_in_all (const char * __first, const char * __last, basic\_string< _CharT, _Traits, _Alloc > &__↵`  
`__outstr, const codecvt< _CharT, char, _State > & __cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool __str_codecvt_out (const _CharT * __first, const _CharT * __last, basic\_string< char, _Traits, _Alloc >`  
`& __outstr, const codecvt< _CharT, char, _State > & __cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool __str_codecvt_out (const _CharT * __first, const _CharT * __last, basic\_string< char, _Traits, _Alloc >`  
`& __outstr, const codecvt< _CharT, char, _State > & __cvt, _State & __state, size_t & __count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool __str_codecvt_out_all (const _CharT * __first, const _CharT * __last, basic\_string< char, _Traits, _Alloc >`  
`& __outstr, const codecvt< _CharT, char, _State > & __cvt)`

- `template<typename _Str>`  
`constexpr _Str __str_concat (typename _Str::value_type const *__lhs, typename _Str::size_type __lhs_len,`  
`typename _Str::value_type const *__rhs, typename _Str::size_type __rhs_len, typename _Str::allocator_type`  
`const &__a)`
- `constexpr size_t __sv_check (size_t __size, size_t __pos, const char *__s)`
- `constexpr size_t __sv_limit (size_t __size, size_t __pos, size_t __off) noexcept`
- `void __terminate () noexcept`
- `void __throw_bad_alloc (void)`
- `void __throw_bad_array_new_length (void)`
- `void __throw_bad_cast (void)`
- `void __throw_bad_exception (void)`
- `void __throw_bad_function_call ()`
- `void __throw_bad_typeid (void)`
- `void __throw_bad_weak_ptr ()`
- `void __throw_domain_error (const char *)`
- `void __throw_future_error (int)`
- `void __throw_invalid_argument (const char *)`
- `void __throw_ios_failure (const char *)`
- `void __throw_ios_failure (const char *, int)`
- `void __throw_length_error (const char *)`
- `void __throw_logic_error (const char *)`
- `void __throw_out_of_range (const char *)`
- `void __throw_out_of_range_fmt (const char *,...)`
- `void __throw_overflow_error (const char *)`
- `void __throw_range_error (const char *)`
- `void __throw_runtime_error (const char *)`
- `void __throw_system_error (int)`
- `void __throw_underflow_error (const char *)`
- `template<typename _Tp>`  
`constexpr to\_chars\_result __to_chars_i (char *__first, char *__last, _Tp __value, int __base=10)`
- `constexpr void __to_wstring_numeric (const char *__s, int __len, wchar_t *__wout)`
- `wstring __to_wstring_numeric (const string &__s)`
- `template<typename _Facet>`  
`const _Facet * __try_use_facet (const locale &__loc)`
- `template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>`  
`constexpr _Cat __tuple_cmp (const _Tp &__t, const _Up &__u, _IndexSeq __indices)`
- `template<typename _Tp, typename _ForwardIterator>`  
`void __uninitialized_construct_buf (_Tp *__first, _Tp *__last, _ForwardIterator __seed)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator __unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵`  
`binary_pred)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __↵`  
`__result, _BinaryPredicate __binary_pred, forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`_BinaryPredicate __binary_pred, input\_iterator\_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>`  
`_ForwardIterator __unique_copy_1 (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, __↵`  
`BinaryPredicate __binary_pred, __true_type)`



- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator __unique_copy_1 (_InputIterator __first, _InputIterator __last, _OutputIterator __↵`  
`result, _BinaryPredicate __binary_pred, __false_type)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator __upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val,`  
`_Compare __comp)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __k)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array<`  
`size_t > __i)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array<`  
`size_t > __j)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t >`  
`__i)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __↵`  
`__n)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __↵`  
`__i)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`

- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy_construct (const _Expr<_Dom, _Tp> &__e, size_t __n, _Array<_Tp> __a)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp>`  
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp>`  
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp>`  
`void __valarray_fill (_Array<_Tp> __a, _Array<size_t> __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Array<_Tp> __a, size_t __n, _Array<bool> __m, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Array<_Tp> __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp &__t)`
- `template<typename _Tp>`  
`_Tp *__valarray_get_storage (size_t)`
- `template<typename _Ta>`  
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta>`  
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `void __valarray_release_memory (void *__p)`
- `template<typename _Tp>`  
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<template<typename> class _Trait, typename _Tp, typename _Up = _Tp>`  
`constexpr _Up __value_or (_Up __def= _Up()) noexcept`

- `bool __verify_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`
- `template<typename _CharT, typename _Outlter>  
_Outlter __write (_Outlter __s, const _CharT *__ws, int __len)`
- `template<typename _CharT>  
ostreambuf\_iterator< _CharT > __write (ostreambuf\_iterator< _CharT > __s, const _CharT *__ws, int __len)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>  
void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>  
void __Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>  
void __Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>  
void __Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>  
void __Array_augmented__bitwise_or (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t`  
`__n)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t`  
`__n)`

- `template<typename _Tp>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented___modulus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented___multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented___multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented___multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented___multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented___multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented___multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, typename... _Args>`  
`constexpr void _Construct (_Tp *__p, _Args &&... __args)`
- `template<typename _T1>`  
`void _Construct_novalue (_T1 *__p)`
- `template<typename _ForwardIterator>`  
`constexpr void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`



- `template<typename _Tp>`  
`constexpr void _Destroy (_Tp * __pointer)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator _Destroy_n (_ForwardIterator __first, _Size __count)`
- `size_t _Fnv_hash_bytes (const void * __ptr, size_t __len, size_t __seed)`
- `size_t _Hash_bytes (const void * __ptr, size_t __len, size_t __seed)`
- `unsigned int _Rb_tree_black_count (const _Rb_tree_node_base * __node, const _Rb_tree_node_base * __root) throw ()`
- `_Rb_tree_node_base * _Rb_tree_decrement (_Rb_tree_node_base * __x) throw ()`
- `_Rb_tree_node_base * _Rb_tree_increment (_Rb_tree_node_base * __x) throw ()`
- `void _Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base * __x, _Rb_tree_node_base * __p, _Rb_tree_node_base & __header) throw ()`
- `_Rb_tree_node_base * _Rb_tree_rebalance_for_erase (_Rb_tree_node_base * const __z, _Rb_tree_node_base & __header) throw ()`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Abs, _Expr, _Dom >, typename _Dom::value_type > abs (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp>`  
`_Tp abs (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Abs, _ValArray, _Tp >, _Tp > abs (const valarray< _Tp > & __v)`
- `constexpr double abs (double __x)`
- `constexpr float abs (float __x)`
- `long abs (long __i)`
- `constexpr long double abs (long double __x)`
- `long long abs (long long __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::abs(declval<double>())) >, _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > abs (simd< _Tp, _Abi > __x)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type acos (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Acos, _Expr, _Dom >, typename _Dom::value_type > acos (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp>`  
`std::complex< _Tp > acos (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Acos, _ValArray, _Tp >, _Tp > acos (const valarray< _Tp > & __v)`
- `constexpr float acos (float __x)`
- `constexpr long double acos (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::acos(declval<double>())) >, _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > acos (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type acosh (_Tp __x)`

- `template<typename _Tp>`  
`std::complex< _Tp > acosh (const std::complex< _Tp > &)`
- `constexpr float acosh (float __x)`
- `constexpr long double acosh (long double __x)`
- `template<typename _Tp, typename _Abi, typename... , typename _R = _Math_return_type_t< decltype(std::acosh(declval<double> ())), <_Tp, _Abi>>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > acosh (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr _Tp * addressof (_Tp &__r) noexcept`
- `template<typename _Tp>`  
`const _Tp * addressof (const _Tp &&)=delete`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Filter>`  
`constexpr _Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate>`  
`constexpr _Filter adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Distance>`  
`constexpr void advance (_InputIterator &__i, _Distance __n)`
- `template<typename _CharT, typename _Distance>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type advance (istreambuf_iterator< _CharT > &__i, _Distance __n)`
- `void * align (size_t __align, size_t __size, void *&__ptr, size_t __space) noexcept`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool all_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool any_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type arg (_Tp __x)`
- `template<typename _Tp>`  
`_Tp arg (const complex< _Tp > &)`
- `template<typename _Tp, typename... _Up>`  
`array (_Tp, _Up...) -> array< enable_if_t<(is_same_v< _Tp, _Up > &&...), _Tp >, 1+sizeof...(Up)>`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type asin (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std:: Asin, _Expr, _Dom >, typename _Dom::value_type > asin (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp>`  
`std::complex< _Tp > asin (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp > &__v)`
- `constexpr float asin (float __x)`
- `constexpr long double asin (long double __x)`
- `template<typename _Tp, typename _Abi, typename... , typename _R = _Math_return_type_t< decltype(std::asin(declval<double>())) , _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > asin (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type asinh (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > asinh (const std::complex< _Tp > &)`
- `constexpr float asinh (float __x)`
- `constexpr long double asinh (long double __x)`
- `template<typename _Tp, typename _Abi, typename... , typename _R = _Math_return_type_t< decltype(std::asinh(declval<double>())) , ←`  
`_Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > asinh (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<size_t _Align, class _Tp>`  
`constexpr _Tp * assume_aligned (_Tp * __ptr) noexcept`
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (_Fn &&__fn, _Args &&... __args)`
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (launch __policy, _Fn &&__fn, _Args &&... __args)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type atan (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`std::complex< _Tp > atan (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > atan (const valarray< _Tp > &__v)`
- `constexpr float atan (float __x)`
- `constexpr long double atan (long double __x)`
- `template<typename _Tp, typename _Abi, typename... , typename _R = _Math_return_type_t< decltype(std::atan(declval<double>())) , _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > atan (simd< _Tp, _Abi > __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type atan2 (_Tp __y, _Up __x)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::atan2(declval<double>()), _Arg2::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > atan2 (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > atan2 (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr float atan2 (float __y, float __x)`
- `constexpr long double atan2 (long double __y, long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::value, double >::type atanh (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > atanh (const std::complex< _Tp > &)`
- `constexpr float atanh (float __x)`
- `constexpr long double atanh (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::atanh(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > atanh (simd< _Tp, _Abi > __x)`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_strong (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > *__i2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > *__i2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > *__i2, memory_order __m1, memory_order __m2) noexcept`

- `template<typename _ITp>`  
`bool atomic_compare_exchange_strong_explicit (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> *__i1, __atomic_val_t<_ITp> __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_weak (atomic<_ITp> *__a, __atomic_val_t<_ITp> *__i1, __atomic_val_t<_ITp> __i2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_weak (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> *__i1, __atomic_val_t<_ITp> __i2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_weak_explicit (atomic<_ITp> *__a, __atomic_val_t<_ITp> *__i1, __atomic_val_t<_ITp> __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_weak_explicit (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> *__i1, __atomic_val_t<_ITp> __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_exchange (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_exchange (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_exchange_explicit (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_exchange_explicit (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_add (atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_add (volatile atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_add_explicit (atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_add_explicit (volatile atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_and (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_and (volatile __atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_and_explicit (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_and_explicit (volatile __atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_or (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_or (volatile __atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_or_explicit (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`

- `template<typename _ITp>`  
`_ITp atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_sub (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_sub (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_sub_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_sub_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_xor (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void atomic_flag_clear (atomic_flag *__a) noexcept`
- `void atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void atomic_init (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void atomic_init (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`bool atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`bool atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_load (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_load_explicit (const atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_load_explicit (const volatile atomic< _ITp > *__a, memory_order __m) noexcept`
- `void atomic_signal_fence (memory_order __m) noexcept`
- `template<typename _ITp>`  
`void atomic_store (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void atomic_store (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`

- `template<typename _ITp>`  
`void atomic_store_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void atomic_store_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void atomic_thread_fence (memory_order __m) noexcept`
- `template<typename _Container>`  
`constexpr back_insert_iterator< _Container > back_inserter (_Container &__x)`
- `template<typename _ForwardIterator>`  
`basic_regex (_ForwardIterator, _ForwardIterator, regex_constants::syntax_option_type= {}) -> basic_regex< typename iterator_traits< _ForwardIterator >::value_type >`
- `template<typename _InputIterator, typename _CharT = typename iterator_traits< _InputIterator >::value_type, typename _Allocator = allocator< _CharT >, typename = _RequireInputIter< _InputIterator >, typename = _RequireAllocator< _Allocator >>`  
`basic_string (_InputIterator, _InputIterator, _Allocator= _Allocator()) -> basic_string< _CharT, char_traits< _CharT >, _Allocator >`
- `template<typename _CharT, typename _Traits, typename _Allocator = allocator< _CharT >, typename = _RequireAllocator< _Allocator >>`  
`basic_string (basic_string_view< _CharT, _Traits >, const _Allocator &= _Allocator()) -> basic_string< _CharT, _Traits, _Allocator >`
- `template<typename _CharT, typename _Traits, typename _Allocator = allocator< _CharT >, typename = _RequireAllocator< _Allocator >>`  
`basic_string (basic_string_view< _CharT, _Traits >, typename basic_string< _CharT, _Traits, _Allocator >::size_type, typename basic_string< _CharT, _Traits, _Allocator >::size_type, const _Allocator &= _Allocator()) -> basic_string< _CharT, _Traits, _Allocator >`
- `template<typename _Operation>`  
`void basic_string< _CharT, _Traits, _Alloc > __resize_and_overwrite (const size_type __n, _Operation __op)`
- `template<typename _Container>`  
`constexpr auto begin (_Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * begin (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto begin (const _Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())`
- `template<class _Tp>`  
`const _Tp * begin (const valarray< _Tp > &__va) noexcept`
- `template<class _Tp>`  
`_Tp * begin (valarray< _Tp > &__va) noexcept`
- `template<class _Tp>`  
`constexpr const _Tp * begin (initializer_list< _Tp > __ils) noexcept`
- `template<typename _Tpa, typename _Tpb>`  
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type beta (_Tpa __a, _Tpb __b)`
- `float betaf (float __a, float __b)`
- `long double betal (long double __a, long double __b)`
- `template<typename _Filter, typename _Tp>`  
`constexpr bool binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare>`  
`constexpr bool binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Func, typename... _BoundArgs>`  
`constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type bind (_Func &&__f, _BoundArgs &&... __args)`

- `template<typename _Result, typename _Func, typename... _BoundArgs>`  
`constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type bind ( _Func &&__f, _BoundArgs &&... __args )`
- `template<typename _Operation, typename _Tp>`  
`binder1st< _Operation > bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp>`  
`binder2nd< _Operation > bind2nd (const _Operation &__fn, const _Tp &__x)`
- `ios_base & boolalpha (ios_base &__base)`
- `template<typename _Callable, typename... _Args>`  
`void call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)`
- `template<typename _Container>`  
`constexpr auto cbegin (const _Container &__cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(__cont))`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type cbrt ( _Tp __x)`
- `constexpr float cbrt (float __x)`
- `constexpr long double cbrt (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::cbrt(std::declval<double>())) , _Tp, _Abi >>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > cbrt (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ceil ( _Tp __x)`
- `constexpr float ceil (float __x)`
- `constexpr long double ceil (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::ceil(std::declval<double>())) , _Tp, _Abi >>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > ceil (simd< _Tp, _Abi > __x)`
- `template<typename _Container>`  
`constexpr auto cend (const _Container &__cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(__cont))`
- `template<typename _Tp>`  
`constexpr const _Tp & clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_1 ( _Tp __k)`
- `float comp_ellint_1f (float __k)`
- `long double comp_ellint_1l (long double __k)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_2 ( _Tp __k)`
- `float comp_ellint_2f (float __k)`
- `long double comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn>`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp_ellint_3 ( _Tp __k, _Tpn __nu)`
- `float comp_ellint_3f (float __k, float __nu)`
- `long double comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tp>`  
`constexpr std::complex< typename __gnu_cxx::__promote< _Tp >::__type > conj ( _Tp __x)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > conj (const complex< _Tp > &)`



- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > const\_pointer\_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename... _Args>`  
`requires (lis_unbounded_array_v<_Tp>) && requires { ::new((void*)0) _Tp(std::declval<_Args>()...); }`  
`constexpr _Tp * construct_at (_Tp * __location, _Args &&... __args) noexcept(noexcept(::new((void *) 0) _Tp(std::declval<_Args>()...)))`
- `template<typename _OI, typename _OI>`  
`constexpr _OI copy (_OI __first, _OI __last, _OI __result)`
- `template<typename _Iter, typename _OIter>`  
`constexpr _OIter copy (_Iter, _Iter, _OIter)`
- `template<typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf\_iterator< _CharT >::type copy`  
`(istreambuf\_iterator< _CharT > __first, istreambuf\_iterator< _CharT > __last, ostreambuf\_iterator< _CharT`  
`> __result)`
- `template<typename _BI1, typename _BI2>`  
`constexpr _BI2 copy\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _BIter1, typename _BIter2>`  
`constexpr _BIter2 copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`constexpr _OIter copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate`  
`__pred)`
- `template<typename _Iter, typename _Size, typename _OIter>`  
`constexpr _OIter copy_n (_Iter, _Size, _OIter)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`constexpr _OutputIterator copy\_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::promote_2< _Tp, _Up >::type copysign (_Tp __x, _Up __y)`
- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>`  
`enable\_if\_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > copysign (const simd< _Tp, _Abi > &__x,`  
`const simd< _Tp, _Abi > &__y)`
- `constexpr float copysign (float __x, float __y)`
- `constexpr long double copysign (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::value, double >::type cos (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::Cos, _Expr, _Dom >, typename _Dom::value_type > cos (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > cos (const complex< _Tp > &)`
- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>`  
`enable\_if\_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > cos (const simd< _Tp, _Abi > &__x)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::Cos, _ValArray, _Tp >, _Tp > cos (const valarray< _Tp > &__v)`
- `constexpr float cos (float __x)`
- `constexpr long double cos (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::value, double >::type cosh (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::Cosh, _Expr, _Dom >, typename _Dom::value_type > cosh (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`

- `template<typename _Tp>`  
`complex< _Tp > cosh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Cosh, _ValArray, _Tp >, _Tp > cosh (const valarray< _Tp > &__v)`
- `constexpr float cosh (float __x)`
- `constexpr long double cosh (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::cosh(declval<double>())), _Tp, _Abi>>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > cosh (simd< _Tp, _Abi > __x)`
- `template<typename _Iter, typename _Tp>`  
`constexpr iterator_traits< _Iter >::difference_type count (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr iterator_traits< _InputIterator >::difference_type count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr iterator_traits< _Iter >::difference_type count_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr iterator_traits< _InputIterator >::difference_type count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Container>`  
`constexpr auto crbegin (const _Container &__cont) noexcept(noexcept(std::rbegin(__cont))) -> decltype(std::rbegin(← __cont))`
- `template<typename _Container>`  
`constexpr auto crend (const _Container &__cont) noexcept(noexcept(std::rend(__cont))) -> decltype(std::rend(← __cont))`
- `exception_ptr current_exception () noexcept`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_if (float __nu, float __x)`
- `long double cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_jf (float __nu, float __x)`
- `long double cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_kf (float __nu, float __x)`
- `long double cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)`
- `float cyl_neumannf (float __nu, float __x)`
- `long double cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Container>`  
`constexpr auto data (_Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * data (_Tp(&__array)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto data (const _Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())`
- `template<typename _Tp>`  
`constexpr const _Tp * data (initializer_list< _Tp > __il) noexcept`
- `ios_base & dec (ios_base &__base)`

- void [declare\\_no\\_pointers](#) (char \*, size\_t)
- void [declare\\_reachable](#) (void \*)
- template<typename \_Tp>  
auto [declval](#) () noexcept -> decltype(\_\_declval< \_Tp >())
- [ios\\_base](#) & [defaultfloat](#) ([ios\\_base](#) & \_\_base)
- template<typename \_InputIterator, typename \_ValT = typename iterator\_traits<\_InputIterator>::value\_type, typename \_Allocator = allocator<\_ValT>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>  
**deque** (\_InputIterator, \_InputIterator, \_Allocator=\_Allocator()) -> deque<\_ValT, \_Allocator >
- template<typename \_ForwardIterator>  
constexpr void **destroy** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_Tp>  
constexpr void **destroy\_at** (\_Tp \*\_\_location)
- template<typename \_ForwardIterator, typename \_Size>  
constexpr \_ForwardIterator **destroy\_n** (\_ForwardIterator \_\_first, \_Size \_\_count)
- template<typename \_InputIterator>  
constexpr [iterator\\_traits](#)<\_InputIterator >::difference\_type [distance](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- ldiv\_t **div** (long \_\_i, long \_\_j) noexcept
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp>  
\_\_shared\_ptr<\_Tp, \_Lp > [dynamic\\_pointer\\_cast](#) (const \_\_shared\_ptr<\_Tp1, \_Lp > &\_\_r) noexcept
- template<typename \_Tp, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Tpp >::\_\_type [ellint\\_1](#) (\_Tp \_\_k, \_Tpp \_\_phi)
- float [ellint\\_1f](#) (float \_\_k, float \_\_phi)
- long double [ellint\\_1l](#) (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Tpp >::\_\_type [ellint\\_2](#) (\_Tp \_\_k, \_Tpp \_\_phi)
- float [ellint\\_2f](#) (float \_\_k, float \_\_phi)
- long double [ellint\\_2l](#) (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_3<\_Tp, \_Tpn, \_Tpp >::\_\_type [ellint\\_3](#) (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float [ellint\\_3f](#) (float \_\_k, float \_\_nu, float \_\_phi)
- long double [ellint\\_3l](#) (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Container>  
constexpr auto [empty](#) (const \_Container &\_\_cont) noexcept(noexcept(\_\_cont.empty())) -> decltype(\_\_cont.empty())
- template<typename \_Tp, size\_t \_Nm>  
constexpr bool [empty](#) (const \_Tp(&)[\_Nm]) noexcept
- template<typename \_Tp>  
constexpr bool [empty](#) ([initializer\\_list](#)<\_Tp > \_\_il) noexcept
- template<typename \_Container>  
constexpr auto [end](#) (\_Container &\_\_cont) noexcept(noexcept(\_\_cont.end())) -> decltype(\_\_cont.end())
- template<typename \_Tp, size\_t \_Nm>  
constexpr \_Tp \* [end](#) (\_Tp(&\_\_arr)[\_Nm]) noexcept
- template<typename \_Container>  
constexpr auto [end](#) (const \_Container &\_\_cont) noexcept(noexcept(\_\_cont.end())) -> decltype(\_\_cont.end())
- template<class \_Tp>  
const \_Tp \* [end](#) (const [valarray](#)<\_Tp > &\_\_va) noexcept
- template<class \_Tp>  
\_Tp \* [end](#) ([valarray](#)<\_Tp > &\_\_va) noexcept
- template<class \_Tp>  
constexpr const \_Tp \* [end](#) ([initializer\\_list](#)<\_Tp > \_\_ils) noexcept
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)<\_CharT, \_Traits > & [endl](#) ([basic\\_ostream](#)<\_CharT, \_Traits > &\_\_os)

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & ends (basic_ostream< _CharT, _Traits > & __os)`
- `template<typename _I1, typename _I2>`  
`constexpr bool equal ( _I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _I1, typename _I2>`  
`constexpr bool equal ( _I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _BinaryPredicate>`  
`constexpr bool equal ( _I1 __first1, _I1 __last1, _I2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2, typename _BinaryPredicate>`  
`constexpr bool equal ( _I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2>`  
`constexpr bool equal ( _I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _Filter, typename _Tp>`  
`constexpr pair< _Filter, _Filter > equal_range ( _Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare>`  
`constexpr pair< _Filter, _Filter > equal_range ( _Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > equal_range ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > equal_range ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>`  
`map< _Key, _Tp, _Compare, _Alloc >::size_type erase_if (map< _Key, _Tp, _Compare, _Alloc > & __cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>`  
`multimap< _Key, _Tp, _Compare, _Alloc >::size_type erase_if (multimap< _Key, _Tp, _Compare, _Alloc > & __cont, _Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
`multiset< _Key, _Compare, _Alloc >::size_type erase_if (multiset< _Key, _Compare, _Alloc > & __cont, _Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
`set< _Key, _Compare, _Alloc >::size_type erase_if (set< _Key, _Compare, _Alloc > & __cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc >::size_type erase_if (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > & __cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc >::size_type erase_if (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > & __cont, _Predicate __pred)`
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`unordered_multiset< _Key, _Hash, _CPred, _Alloc >::size_type erase_if (unordered_multiset< _Key, _Hash, _CPred, _Alloc > & __cont, _Predicate __pred)`
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`unordered_set< _Key, _Hash, _CPred, _Alloc >::size_type erase_if (unordered_set< _Key, _Hash, _CPred, _Alloc > & __cont, _Predicate __pred)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::value, double >::type erf ( _Tp __x)`
- `constexpr float erf (float __x)`
- `constexpr long double erf (long double __x)`

- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::erf(declval<double>()))>, _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > erf (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type erfc (_Tp __x)`
- `constexpr float erfc (float __x)`
- `constexpr long double erfc (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::erfc(declval<double>()))>, _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > erfc (simd< _Tp, _Abi > __x)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _OutputIterator exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type exp (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > exp (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > exp (const valarray< _Tp > &__v)`
- `constexpr float exp (float __x)`
- `constexpr long double exp (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::exp(declval<double>()))>, _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > exp (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type exp2 (_Tp __x)`
- `constexpr float exp2 (float __x)`
- `constexpr long double exp2 (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::exp2(declval<double>()))>, _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > exp2 (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)`
- `float expintf (float __x)`
- `long double expintl (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type expm1 (_Tp __x)`
- `constexpr float expm1 (float __x)`
- `constexpr long double expm1 (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::expm1(declval<double>()))>, _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > expm1 (simd< _Tp, _Abi > __x)`

- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type fabs (_Tp __x)`
- `template<typename _Tp>`  
`_Tp fabs (const std::complex< _Tp > &__z)`
- `constexpr float fabs (float __x)`
- `constexpr long double fabs (long double __x)`
- `template<typename _Tp, typename _Abi, typename... , typename _R = _Math_return_type_t< decltype(std::fabs(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable\_if\_t< is_floating_point_v< _Tp >, _R > fabs (simd< _Tp, _Abi > __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type fdim (_Tp __x, _Up __y)`
- `template<typename _Tp, typename _Abi, typename... , typename _Arg2 = _Extra_argument_type< _Tp, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::fdim(declval<double>()), _Arg2::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable\_if\_t< is_floating_point_v< _Tp >, _R > fdim (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)`
- `constexpr float fdim (float __x, float __y)`
- `constexpr long double fdim (long double __x, long double __y)`
- `template<typename _Filter, typename _Tp>`  
`constexpr void fill (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp>`  
`constexpr _OI fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _OIter, typename _Size, typename _Tp>`  
`constexpr _OIter fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _IIter, typename _Tp>`  
`constexpr _IIter find (_IIter, _IIter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf\_iterator< _CharT >::__type find (istreambuf\_iterator< _CharT > __first, istreambuf\_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr _Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr _Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`

- `template<typename _Iter, typename _Predicate>`  
`constexpr _Iter find_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr _Iter find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `ios_base & fixed (ios_base & __base)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type floor (_Tp __x)`
- `constexpr float floor (float __x)`
- `constexpr long double floor (long double __x)`
- `template<typename _Tp, typename _Abi, typename... , typename _R = _Math_return_type_t< decltype(std::floor(declval<double>())) , _Tp, _Abi >>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > floor (simd< _Tp, _Abi > __x)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & flush (basic_ostream< _CharT, _Traits > & __os)`
- `template<typename _Tp, typename _Up, typename _Vp>`  
`constexpr __gnu_cxx::__promote_3< _Tp, _Up, _Vp >::__type fma (_Tp __x, _Up __y, _Vp __z)`
- `template<typename _Tp, typename _Abi, typename... , typename _Arg2 = _Extra_argument_type< _Tp, _Tp, _Abi >, typename _Arg3 = _Extra_argument_type< _Tp, _Tp, _Abi >, typename _R = _Math_return_type_t< decltype(std::fma(declval<double>(), _Arg2::declval(), _Arg3::declval())), _Tp, _Abi >>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > fma (const simd< _Tp, _Abi > & __x, const typename _Arg2::type & __y, const typename _Arg3::type & __z)`
- `constexpr float fma (float __x, float __y, float __z)`
- `constexpr long double fma (long double __x, long double __y, long double __z)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type fmax (_Tp __x, _Up __y)`
- `template<typename _Tp, typename _Abi, typename... , typename _Arg2 = _Extra_argument_type< _Tp, _Tp, _Abi >, typename _R = _Math_return_type_t< decltype(std::fmax(declval<double>(), _Arg2::declval()), _Tp, _Abi >>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > fmax (const simd< _Tp, _Abi > & __x, const typename _Arg2::type & __y)`
- `constexpr float fmax (float __x, float __y)`
- `constexpr long double fmax (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type fmin (_Tp __x, _Up __y)`
- `template<typename _Tp, typename _Abi, typename... , typename _Arg2 = _Extra_argument_type< _Tp, _Tp, _Abi >, typename _R = _Math_return_type_t< decltype(std::fmin(declval<double>(), _Arg2::declval()), _Tp, _Abi >>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > fmin (const simd< _Tp, _Abi > & __x, const typename _Arg2::type & __y)`
- `constexpr float fmin (float __x, float __y)`
- `constexpr long double fmin (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type fmod (_Tp __x, _Up __y)`
- `template<typename _Tp, typename _Abi, typename... , typename _Arg2 = _Extra_argument_type< _Tp, _Tp, _Abi >, typename _R = _Math_return_type_t< decltype(std::fmod(declval<double>(), _Arg2::declval()), _Tp, _Abi >>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > fmod (const simd< _Tp, _Abi > & __x, const typename _Arg2::type & __y)`
- `constexpr float fmod (float __x, float __y)`
- `constexpr long double fmod (long double __x, long double __y)`

- `template<typename _Iter, typename _Func>`  
`constexpr _Func for_each (_Iter, _Iter, _Func)`
- `template<typename _InputIterator, typename _Function>`  
`constexpr _Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _InputIterator, typename _Size, typename _Function>`  
`constexpr _InputIterator for_each_n (_InputIterator __first, _Size __n, _Function __f)`
- `template<typename _Tp>`  
`constexpr _Tp && forward (typename std::remove_reference< _Tp >::type && __t) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp && forward (typename std::remove_reference< _Tp >::type & __t) noexcept`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &&... > forward_as_tuple (_Elements &&... __args) noexcept`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::__value, int >::__type fpclassify (_Tp __x)`
- `constexpr int fpclassify (double __x)`
- `constexpr int fpclassify (float __x)`
- `constexpr int fpclassify (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::__value, double >::__type frexp (_Tp __x, int * __exp)`
- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>`  
`enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > frexp (const simd< _Tp, _Abi > & __x, __Samesize< int, simd< _Tp, _Abi > > * __exp)`
- `float frexp (float __x, int * __exp)`
- `long double frexp (long double __x, int * __exp)`
- `template<typename _Tp, enable_if_t< __or< __is_standard_integer< _Tp >, is_same< char, remove_cv_t< _Tp > >::value, int > = 0>`  
`constexpr from_chars_result from_chars (const char * __first, const char * __last, _Tp & __value, int __base=10)`
- `from_chars_result from_chars (const char * __first, const char * __last, double & __value, chars_format __fmt=chars_format::general) noexcept`
- `from_chars_result from_chars (const char * __first, const char * __last, float & __value, chars_format __fmt=chars_format::general) noexcept`
- `from_chars_result from_chars (const char * __first, const char * __last, long double & __value, chars_format __fmt=chars_format::general) noexcept`
- `template<typename _Container>`  
`constexpr front_insert_iterator< _Container > front_inserter (_Container & __x)`
- `template<typename _Fn, typename _Signature = __function_guide_t<_Fn, decltype(&_Fn::operator())>>>`  
`function (_Fn) -> function< _Signature >`
- `template<typename _Res, typename... _ArgTypes>`  
`function (_Res(*)(_ArgTypes...)) -> function< _Res(_ArgTypes...)>`
- `const error_category & future_category () noexcept`
- `template<typename _Filter, typename _Generator>`  
`constexpr void generate (_Filter, _Filter, _Generator)`
- `template<typename _ForwardIterator, typename _Generator>`  
`constexpr void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>`  
`_RealType generate_canonical (_UniformRandomNumberGenerator & __g)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`constexpr _OIter generate_n (_OIter, _Size, _Generator)`



- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`constexpr _OutputIterator generate\_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `const error\_category & generic\_category () noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp && get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp & get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp && get (const array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp & get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>`  
`requires ((_Num == 0 && copyable<_It>) || _Num == 1)`  
`constexpr auto get (const subrange< _It, _Sent, _Kind > &__r)`
- `template<size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > && get (const tuple< _Elements... > &&__t) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > & get (const tuple< _Elements... > &__t) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > && get (tuple< _Elements... > &&__t) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > & get (tuple< _Elements... > &__t) noexcept`
- `Catalogs & get\_catalogs ()`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * get\_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept`
- `template<typename _Del, typename _Tp>`  
`_Del * get\_deleter (const shared\_ptr< _Tp > &__p) noexcept`
- `template<typename _MoneyT>`  
`_Get_money< _MoneyT > get\_money (_MoneyT &__mon, bool __intl=false)`
- `new\_handler get\_new\_handler () noexcept`
- `pointer\_safety get\_pointer\_safety () noexcept`
- `template<typename _Tp>`  
`pair< _Tp *, ptrdiff_t > get\_temporary\_buffer (ptrdiff_t __len) noexcept`
- `terminate\_handler get\_terminate () noexcept`
- `template<typename _CharT>`  
`_Get_time< _CharT > get\_time (std::tm *__tmb, const _CharT *__fmt)`
- `unexpected\_handler get\_unexpected () noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &&__is, basic\_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &&__is, basic\_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`

- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str, _CharT __delim)`
- `template<> basic_istream< char > & getline (basic_istream< char > & __in, basic_string< char > & __str, char __delim)`
- `template<> basic_istream< wchar_t > & getline (basic_istream< wchar_t > & __in, basic_string< wchar_t > & __str, wchar_t __delim)`
- `template<typename _Facet>`  
`bool has_facet (const locale & __loc)`
- `template bool has_facet< codecvt< char, char, mbstate_t > > (const locale &)`
- `template bool has_facet< codecvt< wchar_t, char, mbstate_t > > (const locale &)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)`
- `float hermitef (unsigned int __n, float __x)`
- `long double hermitel (unsigned int __n, long double __x)`
- `ios_base & hex (ios_base & __base)`
- `ios_base & hexfloat (ios_base & __base)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type hypot (_Tp __x, _Up __y)`
- `template<typename _Tp, typename _Abi>`  
`_GLIBCXX_SIMD_INTRINSIC simd< _Tp, _Abi > hypot (const simd< _Tp, _Abi > & __x, const simd< _Tp, _Abi > & __y)`
- `constexpr float hypot (float __x, float __y)`
- `constexpr long double hypot (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type ilogb (_Tp __x)`
- `constexpr int ilogb (float __x)`
- `constexpr int ilogb (long double __x)`
- `template<typename _Tp, typename _Abi, typename... , typename _R = _Math_return_type_t< decltype(std::ilogb(declval<double>())) , _Tp, _Abi >>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > ilogb (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote< _Tp >::__type imag (_Tp)`
- `template<typename _Tp>`  
`constexpr _Tp imag (const complex< _Tp > & __z)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare>`  
`constexpr bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare>`  
`constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`

- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`_BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>`  
`constexpr _OutputIterator inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`_BinaryOperation __binary_op, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`  
`constexpr _Tp inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init,`  
`_BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _BidirectionalIterator>`  
`void inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`void inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last,  $\leftrightarrow$`   
`_Compare __comp)`
- `template<typename _Blter>`  
`void inplace\_merge (_Blter, _Blter, _Blter)`
- `template<typename _Blter, typename _Compare>`  
`void inplace\_merge (_Blter, _Blter, _Blter, _Compare)`
- `template<typename _Container>`  
`constexpr insert\_iterator< _Container > insert (_Container &__x, std::__detail::__range_iter_t< _Container >`  
`__i)`
- `ios\_base & internal (ios\_base &__base)`
- `const error\_category & iostream\_category () noexcept`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `constexpr bool is\_eq (partial_ordering __cmp) noexcept`
- `constexpr bool is\_gt (partial_ordering __cmp) noexcept`
- `constexpr bool is\_gteq (partial_ordering __cmp) noexcept`
- `template<typename _RAIter>`  
`constexpr bool is\_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr bool is\_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr bool is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr bool is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter>`  
`constexpr _RAIter is\_heap\_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr _RAIter is\_heap\_until (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator is\_heap\_until (_RandomAccessIterator __first, _RandomAccessIterator __ $\leftrightarrow$`   
`last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator is\_heap\_until (_RandomAccessIterator __first, _RandomAccessIterator __ $\leftrightarrow$`   
`last, _Compare __comp)`
- `constexpr bool is\_lt (partial_ordering __cmp) noexcept`
- `constexpr bool is\_lteq (partial_ordering __cmp) noexcept`
- `constexpr bool is\_neq (partial_ordering __cmp) noexcept`

- `template<typename _Iter, typename _Predicate>`  
`constexpr bool is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr bool is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr bool is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _Filter>`  
`constexpr bool is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr bool is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Filter>`  
`constexpr _Filter is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _CharT>`  
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool isblank (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::__value, bool >::__type isfinite (_Tp)`
- `constexpr bool isfinite (double __x)`
- `constexpr bool isfinite (float __x)`
- `constexpr bool isfinite (long double __x)`
- `template<typename _CharT>`  
`bool isgraph (_CharT __c, const locale &__loc)`

- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(_is_arithmetic< _Tp >::__value && _is_arithmetic< _Up >::__value),`  
`bool >::__type isgreater (_Tp __x, _Up __y)`
- `constexpr bool isgreater (double __x, double __y)`
- `constexpr bool isgreater (float __x, float __y)`
- `constexpr bool isgreater (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(_is_arithmetic< _Tp >::__value && _is_arithmetic< _Up >::__value),`  
`bool >::__type isgreaterequal (_Tp __x, _Up __y)`
- `constexpr bool isgreaterequal (double __x, double __y)`
- `constexpr bool isgreaterequal (float __x, float __y)`
- `constexpr bool isgreaterequal (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< _is_integer< _Tp >::__value, bool >::__type isinf (_Tp)`
- `constexpr bool isinf (double __x)`
- `constexpr bool isinf (float __x)`
- `constexpr bool isinf (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(_is_arithmetic< _Tp >::__value && _is_arithmetic< _Up >::__value),`  
`bool >::__type isless (_Tp __x, _Up __y)`
- `constexpr bool isless (double __x, double __y)`
- `constexpr bool isless (float __x, float __y)`
- `constexpr bool isless (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(_is_arithmetic< _Tp >::__value && _is_arithmetic< _Up >::__value),`  
`bool >::__type islessequal (_Tp __x, _Up __y)`
- `constexpr bool islessequal (double __x, double __y)`
- `constexpr bool islessequal (float __x, float __y)`
- `constexpr bool islessequal (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(_is_arithmetic< _Tp >::__value && _is_arithmetic< _Up >::__value),`  
`bool >::__type islessgreater (_Tp __x, _Up __y)`
- `constexpr bool islessgreater (double __x, double __y)`
- `constexpr bool islessgreater (float __x, float __y)`
- `constexpr bool islessgreater (long double __x, long double __y)`
- `template<typename _CharT>`  
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< _is_integer< _Tp >::__value, bool >::__type isnan (_Tp)`
- `constexpr bool isnan (double __x)`
- `constexpr bool isnan (float __x)`
- `constexpr bool isnan (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< _is_integer< _Tp >::__value, bool >::__type isnormal (_Tp __x)`
- `constexpr bool isnormal (double __x)`
- `constexpr bool isnormal (float __x)`
- `constexpr bool isnormal (long double __x)`
- `template<typename _CharT>`  
`bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool ispunct (_CharT __c, const locale &__loc)`

- `template<typename _CharT>`  
`bool isspace (_CharT __c, const locale &__loc)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(_is_arithmetic< _Tp >::__value && _is_arithmetic< _Up >::__value),`  
`bool >::__type isunordered (_Tp __x, _Up __y)`
- `constexpr bool isunordered (double __x, double __y)`
- `constexpr bool isunordered (float __x, float __y)`
- `constexpr bool isunordered (long double __x, long double __y)`
- `template<typename _CharT>`  
`bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr void iter_swap (_Filter1, _Filter2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _Tp>`  
`_Tp kill_dependency (_Tp __y) noexcept`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)`
- `float laguerref (unsigned int __n, float __x)`
- `long double laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< _is_integer< _Tp >::__value, double >::__type ldexp (_Tp __x, int __exp)`
- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<int, _Tp, _Abi>, typename _R = _Math↔`  
`_return_type_t< decltype(std::ldexp(declval<double>()), _Arg2::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > ldexp (const simd< _Tp,`  
`_Abi > &__x, const typename _Arg2::type &__y)`
- `constexpr float ldexp (float __x, int __exp)`
- `constexpr long double ldexp (long double __x, int __exp)`
- `ios_base & left (ios_base &__base)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __l, _Tp __x)`
- `float legendref (unsigned int __l, float __x)`
- `long double legendrel (unsigned int __l, long double __x)`
- `template<typename _I1, typename _I2>`  
`constexpr bool lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _Compare>`  
`constexpr bool lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`
- `template<typename _I1, typename _I2>`  
`constexpr bool lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _Compare>`  
`constexpr bool lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`
- `template<typename _InputIter1, typename _InputIter2>`  
`constexpr auto lexicographical_compare_three_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __↔`  
`first2, _InputIter2 __last2)`
- `template<typename _InputIter1, typename _InputIter2, typename _Comp>`  
`constexpr auto lexicographical_compare_three_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2,`  
`_InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< _is_integer< _Tp >::__value, double >::__type lgamma (_Tp __x)`
- `constexpr float lgamma (float __x)`

- constexpr long double **lgamma** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::lgamma(declval<double>())) , \_Tp, \_Abi>>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE enable\_if\_t< is\_floating\_point\_v< \_Tp >, \_R > **lgamma** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_InputIterator, typename \_ValT = typename iterator\_traits<\_InputIterator>::value\_type, typename \_Allocator = allocator<\_ValT>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>  
**list** (\_InputIterator, \_InputIterator, \_Allocator= \_Allocator()) -> list< \_ValT, \_Allocator >
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, longlong >::\_\_type **llrint** (\_Tp \_\_x)
- constexpr long long **llrint** (float \_\_x)
- constexpr long long **llrint** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::llrint(declval<double>())) , \_Tp, \_Abi>>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE enable\_if\_t< is\_floating\_point\_v< \_Tp >, \_R > **llrint** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, longlong >::\_\_type **llround** (\_Tp \_\_x)
- constexpr long long **llround** (float \_\_x)
- constexpr long long **llround** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::llround(declval<double>())) , \_Tp, \_Abi>>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE enable\_if\_t< is\_floating\_point\_v< \_Tp >, \_R > **llround** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_L1, typename \_L2, typename... \_L3>  
void **lock** (\_L1 & \_\_l1, \_L2 & \_\_l2, \_L3 &... \_\_l3)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **log** (\_Tp \_\_x)
- template<class \_Dom>  
\_Expr< \_UnClos< struct std::Log, \_Expr, \_Dom >, typename \_Dom::value\_type > **log** (const \_Expr< \_Dom, typename \_Dom::value\_type > & \_\_e)
- template<typename \_Tp>  
**complex**< \_Tp > **log** (const **complex**< \_Tp > &)
- template<typename \_Tp>  
\_Expr< \_UnClos< struct std::Log, \_ValArray, \_Tp >, \_Tp > **log** (const **valarray**< \_Tp > & \_\_v)
- constexpr float **log** (float \_\_x)
- constexpr long double **log** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::log(declval<double>())) , \_Tp, \_Abi>>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE enable\_if\_t< is\_floating\_point\_v< \_Tp >, \_R > **log** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **log10** (\_Tp \_\_x)
- template<class \_Dom>  
\_Expr< \_UnClos< struct std::Log10, \_Expr, \_Dom >, typename \_Dom::value\_type > **log10** (const \_Expr< \_Dom, typename \_Dom::value\_type > & \_\_e)
- template<typename \_Tp>  
**complex**< \_Tp > **log10** (const **complex**< \_Tp > &)
- template<typename \_Tp>  
\_Expr< \_UnClos< struct std::Log10, \_ValArray, \_Tp >, \_Tp > **log10** (const **valarray**< \_Tp > & \_\_v)
- constexpr float **log10** (float \_\_x)
- constexpr long double **log10** (long double \_\_x)

- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log10(declval<double>()))), ←  
_Tp, _Abi>>  
_GLIBCXX_SIMD_ALWAYS_INLINE enable\_if\_t< is_floating_point_v< _Tp >, _R > log10 (simd< _Tp, _Abi >  
__x)`
- `template<typename _Tp>  
constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log1p (_Tp __x)`
- `constexpr float log1p (float __x)`
- `constexpr long double log1p (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log1p(declval<double>()))), ←  
_Tp, _Abi>>  
_GLIBCXX_SIMD_ALWAYS_INLINE enable\_if\_t< is_floating_point_v< _Tp >, _R > log1p (simd< _Tp, _Abi >  
__x)`
- `template<typename _Tp>  
constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log2 (_Tp __x)`
- `constexpr float log2 (float __x)`
- `constexpr long double log2 (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log2(declval<double>()))), _Tp,  
_Abi>>  
_GLIBCXX_SIMD_ALWAYS_INLINE enable\_if\_t< is_floating_point_v< _Tp >, _R > log2 (simd< _Tp, _Abi >  
__x)`
- `template<typename _Tp>  
constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type logb (_Tp __x)`
- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>  
enable\_if\_t< is_floating_point< _Tp >::value, simd< _Tp, _Abi > > logb (const simd< _Tp, _Abi > &__x)`
- `constexpr float logb (float __x)`
- `constexpr long double logb (long double __x)`
- `template<typename _Filter, typename _Tp>  
constexpr _Filter lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare>  
constexpr _Filter lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp>  
constexpr _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>  
constexpr _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, ←  
_Compare __comp)`
- `template<typename _Tp>  
constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type lrint (_Tp __x)`
- `constexpr long lrint (float __x)`
- `constexpr long lrint (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::lrint(declval<double>()))), _Tp,  
_Abi>>  
_GLIBCXX_SIMD_ALWAYS_INLINE enable\_if\_t< is_floating_point_v< _Tp >, _R > lrint (simd< _Tp, _Abi >  
__x)`
- `template<typename _Tp>  
constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type lround (_Tp __x)`
- `constexpr long lround (float __x)`
- `constexpr long lround (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::lround(declval<double>()))), ←  
_Tp, _Abi>>  
_GLIBCXX_SIMD_ALWAYS_INLINE enable\_if\_t< is_floating_point_v< _Tp >, _R > lround (simd< _Tp, _Abi  
> __x)`
- `error\_code make_error_code (errc __e) noexcept`



- `error_code make_error_code (future_errc __errc) noexcept`
- `error_code make_error_code (io_errc __e) noexcept`
- `error_condition make_error_condition (errc __e) noexcept`
- `error_condition make_error_condition (future_errc __errc) noexcept`
- `error_condition make_error_condition (io_errc __e) noexcept`
- `template<typename _Ex>`  
`exception_ptr make_exception_ptr (_Ex __ex) noexcept`
- `template<typename _RAIter>`  
`constexpr void make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iterator>`  
`constexpr move_iterator< _Iterator > make_move_iterator (_Iterator __i)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr _Tp make_obj_using_allocator (const _Alloc &__a, _Args &&... __args)`
- `template<typename _T1, typename _T2>`  
`constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > make_pair (_T1 && __x, _T2 && __y)`
- `template<typename _Iterator>`  
`constexpr reverse_iterator< _Iterator > make_reverse_iterator (_Iterator __i)`
- `template<typename... _Elements>`  
`constexpr tuple< typename __decay_and_strip< _Elements >::__type... > make_tuple (_Elements &&... __args)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less< __iter_key_t< _InputIterator > >, typename _Allocator = allocator< __iter_val_t< _InputIterator > >, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`map (_InputIterator, _InputIterator, _Compare= _Compare(), _Allocator= _Allocator()) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Compare = less< _Key >, typename _Allocator = allocator< pair< const _Key, _Tp > >, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`map (initializer_list< pair< _Key, _Tp > >, _Compare= _Compare(), _Allocator= _Allocator()) -> map< _Key, _Tp, _Compare, _Allocator >`
- `template<typename _Tp>`  
`constexpr const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter>`  
`constexpr _Filter max_element (_Filter, _Filter)`

- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Class>`  
`constexpr _Mem_fn< _Tp _Class::* > mem_fn (_Tp _Class::* __pm) noexcept`
- `template<typename _Ret, typename _Tp>`  
`const_mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp>`  
`mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`const_mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::* __f)(_Arg) const)`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Ret, typename _Tp>`  
`const_mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp>`  
`mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`const_mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::* __f)(_Arg) const)`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::* __f)(_Arg))`
- `void * memchr (void * __s, int __c, size_t __n)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵  
_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵  
_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr const _Tp & min (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & min (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter>`  
`constexpr _Filter min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp & __a, const _Tp & __b)`

- `template<typename _Tp, typename _Compare>`  
`constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< _Tp, _Tp > minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< _Tp, _Tp > minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter>`  
`constexpr pair< _Filter, _Filter > minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr pair< _Filter, _Filter > minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>`  
`enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > modf (const simd< _Tp, _Abi > &__x, simd< _Tp, _Abi > *__iptr)`
- `float modf (float __x, float *__iptr)`
- `long double modf (long double __x, long double *__iptr)`
- `template<typename _II, typename _OI>`  
`constexpr _OI move (_II __first, _II __last, _OI __result)`
- `template<typename _Tp>`  
`constexpr std::remove_reference< _Tp >::type && move (_Tp &&__t) noexcept`
- `template<typename _BI1, typename _BI2>`  
`constexpr _BI2 move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp>`  
`constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > move_if_noexcept (_Tp &__x) noexcept`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less< __iter_key_t< _InputIterator >>, typename _Allocator = allocator< __iter_key_t< _InputIterator >>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`

- multimap** (`_InputIterator`, `_InputIterator`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multimap< __iter←_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**multimap** (`initializer_list` `pair< _Key, _Tp >`, `_Allocator`) -> `multimap< _Key, _Tp, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Compare = less< _Key >, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
**multimap** (`initializer_list` `pair< _Key, _Tp >`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multimap< _Key, _Tp, _Compare, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**multiset** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `multiset< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
**multiset** (`_InputIterator`, `_InputIterator`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**multiset** (`initializer_list` `_Key >`, `_Allocator`) -> `multiset< _Key, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
**multiset** (`initializer_list` `_Key >`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multiset< _Key, _Compare, _Allocator >`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::__value, double >::__type` **nearbyint** (`_Tp __x`)
- `constexpr float` **nearbyint** (`float __x`)
- `constexpr long double` **nearbyint** (`long double __x`)
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::nearbyint(declval<double>())) , _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE` `enable_if_t< is_floating_point_v< _Tp >, _R >` **nearbyint** (`simd< _Tp, _Abi > __x`)
- `template<typename _InputIterator>`  
`constexpr _InputIterator` **next** (`_InputIterator __x`, `typename iterator_traits< _InputIterator >::difference_type __n=1`)
- `template<typename _BidirectionalIterator>`  
`constexpr bool` **next\_permutation** (`_BidirectionalIterator __first`, `_BidirectionalIterator __last`)
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool` **next\_permutation** (`_BidirectionalIterator __first`, `_BidirectionalIterator __last`, `_Compare __comp`)
- `template<typename _BIter>`  
`constexpr bool` **next\_permutation** (`_BIter`, `_BIter`)
- `template<typename _BIter, typename _Compare>`  
`constexpr bool` **next\_permutation** (`_BIter`, `_BIter`, `_Compare`)
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::promote_2< _Tp, _Up >::__type` **nextafter** (`_Tp __x`, `_Up __y`)
- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::nextafter(declval<double>(), _Arg2::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE` `enable_if_t< is_floating_point_v< _Tp >, _R >` **nextafter** (`const simd< _Tp, _Abi > &__x`, `const typename _Arg2::type &__y`)
- `constexpr float` **nextafter** (`float __x`, `float __y`)
- `constexpr long double` **nextafter** (`long double __x`, `long double __y`)

- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type nexttoward ( _Tp __x, long double __y)`
- `constexpr float nexttoward (float __x, long double __y)`
- `constexpr long double nexttoward (long double __x, long double __y)`
- `ios_base & noboolalpha (ios_base & __base)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool none_of ( _Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool none_of ( _InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote< _Tp >::__type norm ( _Tp __x)`
- `template<typename _Tp>`  
`_Tp constexpr norm (const complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr _Tp norm (const complex< _Tp > & __z)`
- `ios_base & noshowbase (ios_base & __base)`
- `ios_base & noshowpoint (ios_base & __base)`
- `ios_base & noshowpos (ios_base & __base)`
- `ios_base & noskipws (ios_base & __base)`
- `template<typename _Predicate>`  
`constexpr unary_negate< _Predicate > not1 (const _Predicate & __pred)`
- `template<typename _Predicate>`  
`constexpr binary_negate< _Predicate > not2 (const _Predicate & __pred)`
- `void notify_all_at_thread_exit (condition_variable &, unique_lock< mutex >)`
- `ios_base & nunitbuf (ios_base & __base)`
- `ios_base & nouppercase (ios_base & __base)`
- `template<typename _RAIter>`  
`constexpr void nth_element ( _RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void nth_element ( _RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr void nth_element ( _RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void nth_element ( _RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `ios_base & oct (ios_base & __base)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__not_equal_to, typename _Dom1::value_type >::result_type > operator!= (const _Expr< _Dom1, typename _Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<typename _StateT>`  
`bool operator!= (const fpos< _StateT > & __lhs, const fpos< _StateT > & __rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > operator!= (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > operator!= (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__modulus, typename _Dom1::value_type >::result_type > operator% (const _Expr< _Dom1, typename _`  
`Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const typename _`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr _ios_Fmtflags operator& (_ios_Fmtflags __a, _ios_Fmtflags __b) noexcept`
- `constexpr _ios_ostate operator& (_ios_ostate __a, _ios_ostate __b) noexcept`
- `constexpr _ios_Openmode operator& (_ios_Openmode __a, _ios_Openmode __b) noexcept`
- `constexpr chars_format operator& (chars_format __lhs, chars_format __rhs) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::`  
`::__bitwise_and, typename _Dom1::value_type >::result_type > operator& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr launch operator& (launch __x, launch __y) noexcept`
- `constexpr memory\_order operator& (memory\_order __m, __memory_order_modifier __mod) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const _Expr< __`  
`Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const __`  
`Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::`  
`::__logical_and, typename _Dom1::value_type >::result_type > operator&& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr const _los_Fmtflags & operator&= (_los_Fmtflags &__a, _los_Fmtflags __b) noexcept`
- `constexpr const _los_losestate & operator&= (_los_losestate &__a, _los_losestate __b) noexcept`
- `constexpr const _los_Openmode & operator&= (_los_Openmode &__a, _los_Openmode __b) noexcept`
- `constexpr chars\_format & operator&= (chars\_format &__lhs, chars\_format __rhs) noexcept`
- `constexpr launch & operator&= (launch &__x, launch __y) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`



- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__multiplies, typename _Dom1::value_type >::result_type > operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__plus, typename _Dom1::value_type >::result_type > operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`



- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &←`  
`__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &←`  
`__lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator+ (const complex< _Tp > & __x)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ←`  
`__fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const typename _Dom←`  
`::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename ←`  
`__fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const valarray< typename`  
`_Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Iterator>`  
`requires requires { { __x.base() + __n } -> same_as<_Iterator>; }`  
`constexpr move_iterator< _Iterator > operator+ (typename move_iterator< _Iterator >::difference_type __n,`  
`const move_iterator< _Iterator > & __x)`
- `template<typename _Iterator>`  
`constexpr reverse_iterator< _Iterator > operator+ (typename reverse_iterator< _Iterator >::difference_type ←`  
`__n, const reverse_iterator< _Iterator > & __x)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename ←`  
`__fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, type-`  
`name _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename ←`  
`__fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, type-`  
`name _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__minus,`  
`typename _Dom1::value_type >::result_type > operator- (const _Expr< _Dom1, typename _Dom1::value_type`  
`> & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator- (const complex< _Tp > & __x)`
- `template<typename _IteratorL, typename _IteratorR>`  
`constexpr auto operator- (const move_iterator< _IteratorL > & __x, const move_iterator< _IteratorR > & __y) ->`  
`decltype(__x.base() - __y.base())`
- `template<typename _IteratorL, typename _IteratorR>`  
`constexpr auto operator- (const reverse_iterator< _IteratorL > & __x, const reverse_iterator< _IteratorR > & __y)`  
`-> decltype(__y.base() - __x.base())`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const typename _Dom←`  
`::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename ←`  
`__fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const valarray< typename`  
`_Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`divides, typename _Dom1::value_type >::result_type > operator/ (const _Expr< _Dom1, typename _Dom1`  
`::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const typename _Dom`  
`::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`less, typename _Dom1::value_type >::result_type > operator< (const _Expr< _Dom1, typename _Dom1::value`  
`type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator< (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,`  
`_Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator< (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &`  
`__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _<`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const typename _Dom<`  
`::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator< (const unique\_ptr< _Tp, _Dp > & __x, const unique\_ptr< _Up, _Ep > & __y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator< (const unique\_ptr< _Tp, _Dp > & __x, nullptr\_t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _<`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const valarray< typename`  
`_Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator< (nullptr\_t, const unique\_ptr< _Tp, _Dp > & __x)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, const error\_code`  
`& __e)`
- `template<typename _Ostream, typename _Tp>`  
`__rvalue_stream_insertion_t< _Ostream, _Tp > operator<< (_Ostream && __os, const _Tp & __x)`
- `template<typename _CharT, typename _Traits, typename _MoneyT>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, _Put_money<`  
`_MoneyT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, _Put_time< _<`  
`CharT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, _Resetiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, _Setbase __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, _Setfill< _CharT`  
`> __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, _Setprecision`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, _Setw __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, basic\_string\_view<`  
`_CharT, _Traits > __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, const`  
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, const`  
`basic\_string< _CharT, _Traits, _Alloc > & __str)`
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > & __os, const complex<`  
`_Tp > & __x)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__shift_left, typename _Dom1::value_type >::result_type > operator<< (const _Expr< _Dom1, typename __`  
`Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const typename __`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template ostream & operator<< (ostream &, _Resetiosflags)`
- `template ostream & operator<< (ostream &, _Setbase)`
- `template ostream & operator<< (ostream &, _Setfill< char >)`
- `template ostream & operator<< (ostream &, _Setiosflags)`
- `template ostream & operator<< (ostream &, _Setprecision)`
- `template ostream & operator<< (ostream &, _Setw)`
- `template ostream & operator<< (ostream &, const complex< double > &)`
- `template ostream & operator<< (ostream &, const complex< float > &)`
- `template ostream & operator<< (ostream &, const complex< long double > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &, const`  
`std::uniform\_int\_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &, const`  
`std::uniform\_real\_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const`  
`binomial\_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const`  
`chi\_squared\_distribution< _RealType > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const`  
`discard\_block\_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const`  
`discrete\_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const`  
`fisher\_f\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const`  
`gamma\_distribution< _RealType > &__x)`

- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`lognormal_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _`  
`UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`negative_binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`poisson_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::bernoulli_distribution &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`student_t_distribution< _RealType > &__x)`

- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template wostream & operator<< (wostream &, _Resetiosflags)`
- `template wostream & operator<< (wostream &, _Setbase)`
- `template wostream & operator<< (wostream &, _Setfill< wchar_t >)`
- `template wostream & operator<< (wostream &, _Setiosflags)`
- `template wostream & operator<< (wostream &, _Setprecision)`
- `template wostream & operator<< (wostream &, _Setw)`
- `template wostream & operator<< (wostream &, const complex< double > &)`
- `template wostream & operator<< (wostream &, const complex< float > &)`
- `template wostream & operator<< (wostream &, const complex< long double > &)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr<`  
`_Tp, _Lp > &__p)`
- `template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr<`  
`_Tp, _Dp > &__p)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _↵`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _↵`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↵`  
`::__less_equal, typename _Dom1::value_type >::result_type > operator<= (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }`  
`constexpr bool operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↵`  
`_Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const typename ↵`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`



- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `strong_ordering operator<=> (const error_code &__lhs, const error_code &__rhs) noexcept`
- `strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr auto operator<=> (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view<`  
`_CharT, _Traits > > __y) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))`
- `template<typename _Tp, typename _Up, _Lock_policy _Lp>`  
`strong_ordering operator<=> (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b)`  
`noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`strong_ordering operator<=> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, size_t _Nm>`  
`constexpr __detail::__synth3way_t< _Tp > operator<=> (const array< _Tp, _Nm > &__a, const array< _Tp,`  
`_Nm > &__b)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr auto operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs) noex-`  
`cept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr auto operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT,`  
`_Traits, _Alloc > &__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))`
- `template<typename _Tp, typename _Alloc>`  
`__detail::__synth3way_t< _Tp > operator<=> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`  
`> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`__detail::__synth3way_t< _Tp > operator<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list<`  
`_Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`__detail::__synth3way_t< _Tp > operator<=> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< pair< const _Key, _Tp > > operator<=> (const map< _Key, _Tp, _Compare, _Alloc`  
`> &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare_three_way_result_t< _Iterator > operator<=> (const move_iterator< _Iterator > &__x,`  
`const move_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare_three_way_result_t< _IteratorL, _IteratorR > operator<=> (const move_iterator< _`  
`IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Tp, three_way_comparable _Seq>`  
`compare_three_way_result_t< _Seq > operator<=> (const queue< _Tp, _Seq > &__x, const queue< _Tp,`  
`_Seq > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare_three_way_result_t< _Iterator, _Iterator > operator<=> (const reverse_iterator< _Iterator`  
`> &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare_three_way_result_t< _IteratorL, _IteratorR > operator<=> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp, three_way_comparable_Seq>`  
`compare_three_way_result_t< _Seq > operator<=> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename... _Tps, typename... _Ups>`  
`requires (sizeof...( _Tps) == sizeof...( _Ups)) && (requires { typename __detail::__synth3way_t< _Tps, _Ups>; } && ...)`  
`constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > operator<=> (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`requires three_way_comparable_with<typename unique_ptr< _Tp, _Dp>::pointer, typename unique_ptr< _Up, _Ep>::pointer>`  
`constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > operator<=> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`requires three_way_comparable<typename unique_ptr< _Tp, _Dp>::pointer>`  
`constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > operator<=> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp > operator<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `constexpr strong_ordering operator<=> (monostate, monostate) noexcept`
- `template<typename _T1, typename _T2>`  
`constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept`
- `bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept`
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _Bi_iter, typename _Alloc>`  
`bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator== (basic_string_view< _CharT, _Traits > __x, type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__equal_to, typename _Dom1::value_type >::result_type > operator== (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`



- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _Tp, typename _Alloc>`  
`bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _StateT>`  
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _Res, typename... _Args>`  
`bool operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _CharT, typename _Traits>`  
`bool operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _Tp, typename _Alloc>`  
`bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Iterator>`  
`constexpr bool operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _RealType>`  
`bool operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename... _Tps, typename... _Ups>`  
`requires (sizeof...( _Tps) == sizeof...( _Ups)) && (requires (const _Tps& __t, const _Ups& __u) { { __t == __u } -> __detail::__boolean_testable; } && ...)`  
`constexpr bool operator== (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `constexpr bool operator== (monostate, monostate) noexcept`
- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
`bool operator== (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`greater, typename _Dom1::value_type >::result_type > operator> (const _Expr< _Dom1, typename _Dom1`  
`::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }`  
`constexpr bool operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`

- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__greater_equal, typename _Dom1::value_type >::result_type > operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>=` (const  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator>=` (const `unique_ptr< _Tp, _Dp >` &\_\_x, const `unique_ptr< _Up, _Ep >` &\_\_y)
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator>=` (const `unique_ptr< _Tp, _Dp >` &\_\_x, `nullptr_t`)
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>=` (const  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Dp>`  
`bool operator>=` (`nullptr_t`, const `unique_ptr< _Tp, _Dp >` &\_\_x)
- `template<typename _Istream, typename _Tp>`  
`_rvalue_stream_extraction_t< _Istream, _Tp > operator>>` (`_Istream &&__is, _Tp &&__x`)
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string<`  
`_CharT, _Traits, _Alloc, _Base > &__str`)
- `template<typename _CharT, typename _Traits, typename _MoneyT>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, _Get_money< _↵`  
`MoneyT > __f`)
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, _Get_time< _CharT`  
`> __f`)
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f`)
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, _Setbase __f`)
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, _Setfill< _CharT >`  
`__f`)
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, _Setiosflags __f`)
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, _Setprecision __f`)
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, _Setw __f`)
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, basic_string< _↵`  
`CharT, _Traits, _Alloc > &__str`)
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream< _CharT, _Traits > &__is, complex< _Tp >`  
`&__x`)
- `template<>` `basic_istream< char > & operator>>` (`basic_istream< char > &__is, basic_string< char > &__↵`  
`__str`)
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>>` (const `_Expr< _↵`  
`Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>>` (const `_Expr< _↵`  
`Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__shift_right, typename _Dom1::value_type >::result_type > operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template istream & operator>> (istream &, _Resetiosflags)`
- `template istream & operator>> (istream &, _Setbase)`
- `template istream & operator>> (istream &, _Setfill< char >)`
- `template istream & operator>> (istream &, _Setiosflags)`
- `template istream & operator>> (istream &, _Setprecision)`
- `template istream & operator>> (istream &, _Setw)`
- `template istream & operator>> (istream &, complex< double > &)`
- `template istream & operator>> (istream &, complex< float > &)`
- `template istream & operator>> (istream &, complex< long double > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &, std::uniform\_int\_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &, std::uniform\_real\_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, binomial\_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, chi\_squared\_distribution< _RealType > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, discard\_block\_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, discrete\_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, fisher\_f\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, gamma\_distribution< _RealType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, linear\_congruential\_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, lognormal\_distribution< _RealType > &__x)`

- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`  
`&__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`negative_binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`poisson_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::bernoulli_distribution &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`student_t_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template wistream & operator>> (wistream &, _Resetiosflags)`
- `template wistream & operator>> (wistream &, _Setbase)`
- `template wistream & operator>> (wistream &, _Setfill< wchar_t >)`
- `template wistream & operator>> (wistream &, _Setiosflags)`
- `template wistream & operator>> (wistream &, _Setprecision)`
- `template wistream & operator>> (wistream &, _Setw)`



- template `wistream` & **operator**>> (`wistream` &, `complex`< double > &)
- template `wistream` & **operator**>> (`wistream` &, `complex`< float > &)
- template `wistream` & **operator**>> (`wistream` &, `complex`< long double > &)
- constexpr `_los_Fmtflags` **operator**<sup>^</sup> (`_los_Fmtflags` \_\_a, `_los_Fmtflags` \_\_b) noexcept
- constexpr `_los_losestate` **operator**<sup>^</sup> (`_los_losestate` \_\_a, `_los_losestate` \_\_b) noexcept
- constexpr `_los_Openmode` **operator**<sup>^</sup> (`_los_Openmode` \_\_a, `_los_Openmode` \_\_b) noexcept
- constexpr `chars_format` **operator**<sup>^</sup> (`chars_format` \_\_lhs, `chars_format` \_\_rhs) noexcept
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_xor, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename  
\_\_fun< struct std::\_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **operator**<sup>^</sup> (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_e, const `valarray`< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_xor, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, type-  
name \_\_fun< struct std::\_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **operator**<sup>^</sup> (const \_Expr<  
\_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom1, class \_Dom2>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_xor, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< struct std::  
\_\_bitwise\_xor, typename \_Dom1::value\_type >::result\_type > **operator**<sup>^</sup> (const \_Expr< \_Dom1, typename  
\_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_xor, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, type-  
name \_\_fun< struct std::\_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **operator**<sup>^</sup> (const type-  
name \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_xor, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename  
\_\_fun< struct std::\_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **operator**<sup>^</sup> (const `valarray`<  
typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- constexpr `launch` **operator**<sup>^</sup> (`launch` \_\_x, `launch` \_\_y) noexcept
- constexpr const `_los_Fmtflags` & **operator**<sup>=</sup> (`_los_Fmtflags` &\_\_a, `_los_Fmtflags` \_\_b) noexcept
- constexpr const `_los_losestate` & **operator**<sup>=</sup> (`_los_losestate` &\_\_a, `_los_losestate` \_\_b) noexcept
- constexpr const `_los_Openmode` & **operator**<sup>=</sup> (`_los_Openmode` &\_\_a, `_los_Openmode` \_\_b) noexcept
- constexpr `chars_format` & **operator**<sup>=</sup> (`chars_format` &\_\_lhs, `chars_format` \_\_rhs) noexcept
- constexpr `launch` & **operator**<sup>=</sup> (`launch` &\_\_x, `launch` \_\_y) noexcept
- constexpr `_los_Fmtflags` **operator**| (`_los_Fmtflags` \_\_a, `_los_Fmtflags` \_\_b) noexcept
- constexpr `_los_losestate` **operator**| (`_los_losestate` \_\_a, `_los_losestate` \_\_b) noexcept
- constexpr `_los_Openmode` **operator**| (`_los_Openmode` \_\_a, `_los_Openmode` \_\_b) noexcept
- constexpr `chars_format` **operator**| (`chars_format` \_\_lhs, `chars_format` \_\_rhs) noexcept
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_or, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename  
\_\_fun< struct std::\_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **operator**| (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_e, const `valarray`< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_or, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename  
\_\_fun< struct std::\_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **operator**| (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom1, class \_Dom2>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_or, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< struct std::  
\_\_bitwise\_or, typename \_Dom1::value\_type >::result\_type > **operator**| (const \_Expr< \_Dom1, typename \_Dom1::  
value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_or, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename  
\_\_fun< struct std::\_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **operator**| (const typename \_Dom::  
value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > operator| (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr launch operator| (launch __x, launch __y) noexcept`
- `constexpr memory\_order operator| (memory\_order __m, __memory_order_modifier __mod) noexcept`
- `constexpr const _los_Fmtflags & operator|= (_los_Fmtflags &__a, _los_Fmtflags __b) noexcept`
- `constexpr const _los_losestate & operator|= (_los_losestate &__a, _los_losestate __b) noexcept`
- `constexpr const _los_Openmode & operator|= (_los_Openmode &__a, _los_Openmode __b) noexcept`
- `constexpr chars\_format & operator|= (chars\_format &__lhs, chars\_format __rhs) noexcept`
- `constexpr launch & operator|= (launch &__x, launch __y) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__logical_or, typename _Dom::value_type >::result_type > operator|| (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__logical_or, typename _Dom::value_type >::result_type > operator|| (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__logical_or, typename _Dom1::value_type >::result_type > operator|| (const _Expr< _Dom1, typename _`  
`Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__logical_or, typename _Dom::value_type >::result_type > operator|| (const typename _`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__logical_or, typename _Dom::value_type >::result_type > operator|| (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr _los_Fmtflags operator~ (_los_Fmtflags __a) noexcept`
- `constexpr _los_losestate operator~ (_los_losestate __a) noexcept`
- `constexpr _los_Openmode operator~ (_los_Openmode __a) noexcept`
- `constexpr chars\_format operator~ (chars\_format __fmt) noexcept`
- `constexpr launch operator~ (launch __x) noexcept`
- `template<typename _Fun, typename _Signature = __function_guide_t< _Fun, decltype(&_Fun::operator())>>`  
`packaged_task (_Fun) -> packaged_task< _Signature >`
- `template<typename _Res, typename... _ArgTypes>`  
`packaged_task (_Res*)(_ArgTypes...) -> packaged_task< _Res(_ArgTypes...)>`
- `template<typename _RAIter>`  
`constexpr void partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr void partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random`  
`AccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random`  
`AccessIterator __last, _Compare __comp)`
- `template<typename _Iter, typename _RAIter>`  
`constexpr _RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`



- `template<typename _Iter, typename _RAIter, typename _Compare>`  
`constexpr _RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _BIter, typename _Predicate>`  
`constexpr _BIter partition (_BIter, _BIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate>`  
`constexpr pair< _OIter1, _OIter2 > partition_copy (_Iter, _Iter, _OIter1, _OIter2, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>`  
`constexpr pair< _OutputIterator1, _OutputIterator2 > partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate>`  
`constexpr _Filter partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp>`  
`complex< _Tp > polar (const _Tp &, const _Tp &= _Tp(0))`
- `template<typename _RAIter>`  
`constexpr void pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type pow (_Tp __x, _Up __y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > pow (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<typename _Tp>`  
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`

- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp>`  
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > pow (const complex< _Tp > &, int)`
- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::pow(declval<double>(), _Arg2::declval())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > pow (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > pow (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > pow (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr float pow (float __x, float __y)`
- `constexpr long double pow (long double __x, long double __y)`
- `template<typename _BidirectionalIterator>`  
`constexpr _BidirectionalIterator prev (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator>`  
`constexpr bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Blter>`  
`constexpr bool prev_permutation (_Blter, _Blter)`
- `template<typename _Blter, typename _Compare>`  
`constexpr bool prev_permutation (_Blter, _Blter, _Compare)`
- `template<typename _Compare, typename _Container, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>`  
`priority_queue (_Compare, _Container) -> priority_queue< typename _Container::value_type, _Container, _Compare >`

- `template<typename _Compare, typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>`  
**priority\_queue** (`_Compare`, `_Container`, `_Allocator`) -> `priority_queue< typename _Container::value_type, _Container, _Compare >`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Compare = less<_ValT>, typename _Container = vector<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>`  
**priority\_queue** (`_InputIterator`, `_InputIterator`, `_Compare=_Compare()`, `_Container=_Container()`) -> `priority_queue< _ValT, _Container, _Compare >`
- `template<typename _Tp>`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type >` **proj** (`_Tp __x`)
- `template<typename _Tp>`  
`std::complex< _Tp >` **proj** (`const std::complex< _Tp > &`)
- `template<typename _Arg, typename _Result>`  
`pointer_to_unary_function< _Arg, _Result >` **ptr\_fun** (`_Result(*__x)(_Arg)`)
- `template<typename _Arg1, typename _Arg2, typename _Result>`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result >` **ptr\_fun** (`_Result(*__x)(_Arg1, _Arg2)`)
- `template<typename _RAIter>`  
`constexpr void` **push\_heap** (`_RAIter, _RAIter`)
- `template<typename _RAIter, typename _Compare>`  
`constexpr void` **push\_heap** (`_RAIter, _RAIter, _Compare`)
- `template<typename _RandomAccessIterator>`  
`constexpr void` **push\_heap** (`_RandomAccessIterator __first, _RandomAccessIterator __last`)
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void` **push\_heap** (`_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp`)
- `template<typename _MoneyT>`  
`_Put_money< _MoneyT >` **put\_money** (`const _MoneyT &__mon, bool __intl=false`)
- `template<typename _CharT>`  
`_Put_time< _CharT >` **put\_time** (`const std::tm * __tmb, const _CharT * __fmt`)
- `template<typename _Container, typename = _RequireNotAllocator<_Container>>`  
**queue** (`_Container`) -> `queue< typename _Container::value_type, _Container >`
- `template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>`  
**queue** (`_Container, _Allocator`) -> `queue< typename _Container::value_type, _Container >`
- `template<typename _RAIter>`  
`void` **random\_shuffle** (`_RAIter, _RAIter`)
- `template<typename _RAIter, typename _Generator>`  
`void` **random\_shuffle** (`_RAIter, _RAIter, _Generator &&`)
- `template<typename _RandomAccessIterator>`  
`void` **random\_shuffle** (`_RandomAccessIterator __first, _RandomAccessIterator __last`)
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`void` **random\_shuffle** (`_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand`)
- `template<typename _Container>`  
`constexpr auto` **rbegin** (`_Container &__cont`) `noexcept(noexcept(__cont.rbegin()))` -> `decltype(__cont.rbegin())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr` **reverse\_iterator**< `_Tp *` > **rbegin** (`_Tp(&__arr)[_Nm]`) `noexcept`
- `template<typename _Container>`  
`constexpr auto` **rbegin** (`const _Container &__cont`) `noexcept(noexcept(__cont.rbegin()))` -> `decltype(__cont.rbegin())`
- `template<typename _Tp>`  
`constexpr` **reverse\_iterator**< `const _Tp *` > **rbegin** (`initializer_list< _Tp > __il`) `noexcept`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote< _Tp >::__type` **real** (`_Tp __x`)

- `template<typename _Tp>`  
`constexpr _Tp real (const complex< _Tp > &__z)`
- `template<typename _InputIterator>`  
`constexpr iterator_traits< _InputIterator >::value_type reduce (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp reduce (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Tp>`  
`reference_wrapper (_Tp &) -> reference_wrapper< _Tp >`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > reinterpret_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type remainder (_Tp __x, _Up __y)`
- `template<typename _Tp, typename _Abi, typename... , typename _Arg2 = _Extra_argument_type< _Tp, _Tp, _Abi>, typename _R = _↔  
Math_return_type_t< decltype(std::remainder(declval<double>(), _Arg2::declval())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > remainder (const simd<  
_Tp, _Abi > &__x, const typename _Arg2::type &__y)`
- `constexpr float remainder (float __x, float __y)`
- `constexpr long double remainder (long double __x, long double __y)`
- `template<typename _Filter, typename _Tp>`  
`constexpr _Filter remove (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Tp>`  
`constexpr _OIter remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,  
const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`constexpr _OIter remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result,  
_Predicate __pred)`
- `template<typename _Filter, typename _Predicate>`  
`constexpr _Filter remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Up>`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type remquo (_Tp __x, _Up __y, int *__pquo)`
- `template<typename _Tp, typename _Abi, typename... , typename _Arg2 = _Extra_argument_type< _Tp, _Tp, _Abi>, typename _Arg3 = _↔  
Extra_argument_type<int*, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::remquo(declval<double>(), _Arg2::declval(),  
_Arg3::declval())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > remquo (const simd< _Tp,  
_Abi > &__x, const typename _Arg2::type &__y, const typename _Arg3::type &__z)`
- `float remquo (float __x, float __y, int *__pquo)`
- `long double remquo (long double __x, long double __y, int *__pquo)`
- `template<typename _Container>`  
`constexpr auto rend (_Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse_iterator< _Tp * > rend (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto rend (const _Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())`

- `template<typename _Tp>`  
`constexpr reverse\_iterator< const _Tp * > rend (initializer\_list< _Tp > __il) noexcept`
- `template<typename _Filter, typename _Tp>`  
`constexpr void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Tp>`  
`constexpr _OIter replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator replace\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>`  
`constexpr _OutputIterator replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp>`  
`constexpr _OIter replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`constexpr void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp>`  
`constexpr void replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)↵`
- `_Resetiosflags resetiosflags (ios\_base::fmtflags __mask)`
- `void rethrow\_exception (exception\_ptr)`
- `template<typename _Ex>`  
`void rethrow\_if\_nested (const _Ex &__ex)`
- `template<typename _Tp>`  
`void return\_temporary\_buffer (_Tp *__p)`
- `template<typename _BidirectionalIterator>`  
`constexpr void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BIter>`  
`constexpr void reverse (_BIter, _BIter)`
- `template<typename _BidirectionalIterator, typename _OutputIterator>`  
`constexpr _OutputIterator reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)↵`
- `template<typename _BIter, typename _OIter>`  
`constexpr _OIter reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type riemann\_zeta (_Tp __s)`
- `float riemann\_zetaf (float __s)`
- `long double riemann\_zetal (long double __s)`
- `ios\_base & right (ios\_base &__base)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type rint (_Tp __x)`
- `constexpr float rint (float __x)`
- `constexpr long double rint (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::rint(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable\_if\_t< is_floating_point_v< _Tp >, _R > rint (simd< _Tp, _Abi > __x)`
- `template<typename _Filter>`  
`constexpr _Filter rotate (_Filter, _Filter, _Filter)`

- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _Filter, typename _OIter>`  
`constexpr _OIter rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _ForwardIterator, typename _OutputIterator>`  
`constexpr _OutputIterator rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type round (_Tp __x)`
- `constexpr float round (float __x)`
- `constexpr long double round (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::round(declval<double>())), __<_Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > round (simd< _Tp, _Abi > __x)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomBitGenerator>`  
`_SampleIterator sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type scalbn (_Tp __x, long __ex)`
- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<long, _Tp, _Abi>, typename _R = __<_Math_return_type_t< decltype(std::scalbn(declval<double>()), _Arg2::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > scalbn (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)`
- `constexpr float scalbn (float __x, long __ex)`
- `constexpr long double scalbn (long double __x, long __ex)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type scalbn (_Tp __x, int __ex)`
- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<int, _Tp, _Abi>, typename _R = _Math__<_return_type_t< decltype(std::scalbn(declval<double>()), _Arg2::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > scalbn (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)`
- `constexpr float scalbn (float __x, int __ex)`
- `constexpr long double scalbn (long double __x, int __ex)`
- `ios_base & scientific (ios_base &__base)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`constexpr _ForwardIterator search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _Filter, typename _Size, typename _Tp>`  
`constexpr _Filter search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate>`  
`constexpr _Filter search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`

- `template<typename _ForwardIterator, typename _Integer, typename _Tp>`  
`constexpr _ForwardIterator search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`constexpr _ForwardIterator search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator\_traits< _InputIterator >::value_type, less< typename iterator\_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator\_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator\_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator\_traits< _InputIterator >::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`set (initializer\_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`set (initializer\_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `new\_handler set\_new\_handler (new\_handler) throw ()`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `terminate\_handler set\_terminate (terminate\_handler) noexcept`



- [unexpected\\_handler](#) [set\\_unexpected](#) ([unexpected\\_handler](#)) noexcept
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter set\_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter set\_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵  
_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵  
_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `_Setbase setbase (int __base)`
- `template<typename _CharT>`  
`_Setfill< _CharT > setfill (_CharT __c)`
- `_Setiosflags setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision setprecision (int __n)`
- `_Setw setw (int __n)`
- `template<typename _Tp, typename _Del>`  
`shared\_ptr (unique\_ptr< _Tp, _Del >) -> shared\_ptr< _Tp >`
- `template<typename _Tp>`  
`shared\_ptr (weak\_ptr< _Tp >) -> shared\_ptr< _Tp >`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator shift\_left (_ForwardIterator __first, _ForwardIterator __last, typename iterator\_traits<  
_ForwardIterator >::difference_type __n)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator shift\_right (_ForwardIterator __first, _ForwardIterator __last, typename iterator\_traits<  
_ForwardIterator >::difference_type __n)`
- `ios_base & showbase (ios_base & __base)`
- `ios_base & showpoint (ios_base & __base)`
- `ios_base & showpos (ios_base & __base)`
- `template<typename _RAlter, typename _UGenerator>`  
`void shuffle (_RAlter, _RAlter, _UGenerator &&)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>`  
`void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator  
&& __g)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type signbit (_Tp __x)`
- `constexpr bool signbit (double __x)`
- `constexpr bool signbit (float __x)`
- `constexpr bool signbit (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sin (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::Sin, _Expr, _Dom >, typename _Dom::value_type > sin (const _Expr< _Dom,  
typename _Dom::value_type > & __e)`
- `template<typename _Tp>`  
`complex< _Tp > sin (const complex< _Tp > &)`
- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>`  
`enable\_if\_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > sin (const simd< _Tp, _Abi > & __x)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::Sin, _ValArray, _Tp >, _Tp > sin (const valarray< _Tp > & __v)`
- `constexpr float sin (float __x)`



- constexpr long double **sin** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **sinh** (\_Tp \_\_x)
- template<class \_Dom>  
\_Expr< \_UnClos< struct std::\_Sinh, \_Expr, \_Dom >, typename \_Dom::value\_type > **sinh** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp>  
**complex**< \_Tp > **sinh** (const **complex**< \_Tp > &)
- template<typename \_Tp>  
\_Expr< \_UnClos< struct std::\_Sinh, \_ValArray, \_Tp >, \_Tp > **sinh** (const **valarray**< \_Tp > &\_\_v)
- constexpr float **sinh** (float \_\_x)
- constexpr long double **sinh** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::sinh(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE **enable\_if\_t**< is\_floating\_point\_v< \_Tp >, \_R > **sinh** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Container>  
constexpr auto **size** (const \_Container &\_\_cont) noexcept(noexcept(\_\_cont.size())) -> decltype(\_\_cont.size())
- template<typename \_Tp, size\_t \_Nm>  
constexpr size\_t **size** (const \_Tp(&)[\_Nm]) noexcept
- **ios\_base** & **skipws** (**ios\_base** &\_\_base)
- template<typename \_RAlter>  
constexpr void **sort** (\_RAlter, \_RAlter)
- template<typename \_RAlter, typename \_Compare>  
constexpr void **sort** (\_RAlter, \_RAlter, \_Compare)
- template<typename \_RandomAccessIterator>  
constexpr void **sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)
- template<typename \_RandomAccessIterator, typename \_Compare>  
constexpr void **sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Compare \_\_comp)
- template<typename \_RAlter>  
constexpr void **sort\_heap** (\_RAlter, \_RAlter)
- template<typename \_RAlter, typename \_Compare>  
constexpr void **sort\_heap** (\_RAlter, \_RAlter, \_Compare)
- template<typename \_RandomAccessIterator>  
constexpr void **sort\_heap** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)
- template<typename \_RandomAccessIterator, typename \_Compare>  
constexpr void **sort\_heap** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Compare \_\_comp)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_bessel** (unsigned int \_\_n, \_Tp \_\_x)
- float **sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **sph\_neumannl** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **sqrt** (\_Tp \_\_x)

- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Sqrt, _Expr, _Dom >, typename _Dom::value_type > sqrt (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > sqrt (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Sqrt, _ValArray, _Tp >, _Tp > sqrt (const valarray< _Tp > &__v)`
- `constexpr float sqrt (float __x)`
- `constexpr long double sqrt (long double __x)`
- `template<typename _Tp, typename _Abi, typename... , typename _R = _Math_return_type_t< decltype(std::sqrt(declval<double>())), _Tp,`  
`_Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > sqrt (simd< _Tp, _Abi >`  
`__x)`
- `template<typename _Container>`  
`constexpr auto ssize (const _Container &__cont) noexcept(noexcept(__cont.size())) -> common_type_t<`  
`ptrdiff_t, make_signed_t< decltype(__cont.size())>> >`
- `template<typename _Tp, ptrdiff_t _Num>`  
`constexpr ptrdiff_t ssize (const _Tp(&)[_Num]) noexcept`
- `template<typename _Blter, typename _Predicate>`  
`_Blter stable_partition (_Blter, _Blter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`_ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`void stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Container, typename = _RequireNotAllocator< _Container>>`  
`stack (_Container) -> stack< typename _Container::value_type, _Container >`
- `template<typename _Container, typename _Allocator, typename = _RequireNotAllocator< _Container>>`  
`stack (_Container, _Allocator) -> stack< typename _Container::value_type, _Container >`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `double stod (const string &__str, size_t * __idx=0)`
- `double stod (const wstring &__str, size_t * __idx=0)`
- `float stof (const string &__str, size_t * __idx=0)`
- `float stof (const wstring &__str, size_t * __idx=0)`
- `int stoi (const string &__str, size_t * __idx=0, int __base=10)`
- `int stoi (const wstring &__str, size_t * __idx=0, int __base=10)`
- `long stol (const string &__str, size_t * __idx=0, int __base=10)`
- `long stol (const wstring &__str, size_t * __idx=0, int __base=10)`
- `long double stold (const string &__str, size_t * __idx=0)`
- `long double stold (const wstring &__str, size_t * __idx=0)`
- `long long stoll (const string &__str, size_t * __idx=0, int __base=10)`
- `long long stoll (const wstring &__str, size_t * __idx=0, int __base=10)`
- `unsigned long stoul (const string &__str, size_t * __idx=0, int __base=10)`
- `unsigned long stoul (const wstring &__str, size_t * __idx=0, int __base=10)`
- `unsigned long long stoull (const string &__str, size_t * __idx=0, int __base=10)`
- `unsigned long long stoull (const wstring &__str, size_t * __idx=0, int __base=10)`

- `char * strchr (char *__s, int __n)`
- `char * strpbrk (char *__s1, const char *__s2)`
- `char * strchr (char *__s, int __n)`
- `char * strstr (char *__s1, const char *__s2)`
- `template<typename _Ch_type, typename _Rx_traits>  
void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs) noexcept`
- `template<typename _Bi_iter, typename _Alloc>  
void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
void swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
void swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc>  
void swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp>  
constexpr Require< __not< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > swap (_Tp &__a, _Tp &__b) noexcept(__and< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)`
- `template<typename _Tp>  
requires (! __is_tuple_like< _Tp >::value) && is_move_constructible_v< _Tp > && is_move_assignable_v< _Tp >  
constexpr void swap (_Tp &__a, _Tp &__b) noexcept(conditional /*/ is_nothrow_move_assignable< _Tp > >`
- `template<typename _Tp, size_t _Nm>  
requires is_swappable_v< _Tp >  
constexpr void swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(conditional /*/`
- `template<typename _Tp, size_t _Nm>  
constexpr __enable_if_t< __is_swappable< _Tp >::value > swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm])  
noexcept(__is_nothrow_swappable< _Tp >::value)`
- `template<typename _Tp, std::size_t _Nm>  
__enable_if_t<! __array_traits< _Tp, _Nm >::is_swappable::value > swap (array< _Tp, _Nm > &__a, array< _Tp, _Nm > &__b)=delete`
- `template<typename _Tp, std::size_t _Nm>  
constexpr __enable_if_t< __array_traits< _Tp, _Nm >::is_swappable::value > swap (array< _Tp, _Nm > &__a, array< _Tp, _Nm > &__b) noexcept(noexcept(__a.swap(__b)))`
- `template<class _CharT, class _Traits>  
void swap (basic_filebuf< _CharT, _Traits > &__x, basic_filebuf< _CharT, _Traits > &__y)`
- `template<class _CharT, class _Traits>  
void swap (basic_fstream< _CharT, _Traits > &__x, basic_fstream< _CharT, _Traits > &__y)`
- `template<class _CharT, class _Traits>  
void swap (basic_ifstream< _CharT, _Traits > &__x, basic_ifstream< _CharT, _Traits > &__y)`
- `template<class _CharT, class _Traits, class _Allocator>  
void swap (basic_istream< _CharT, _Traits, _Allocator > &__x, basic_istream< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits>  
void swap (basic_ofstream< _CharT, _Traits > &__x, basic_ofstream< _CharT, _Traits > &__y)`
- `template<class _CharT, class _Traits, class _Allocator>  
void swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits, _Allocator > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept(conditional /*/`

- `template<class _CharT, class _Traits, class _Allocator>`  
`void swap(basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _CharT, class _Traits, class _Allocator>`  
`void swap(basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _Allocator > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`void swap(deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `void swap(exception_ptr &__lhs, exception_ptr &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`void swap(forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`
- `template<typename _Res, typename... _Args>`  
`void swap(function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y) noexcept`
- `template<typename _Tp, typename _Alloc>`  
`void swap(list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`void swap(map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`void swap(multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`void swap(multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Res, typename... _ArgTypes>`  
`void swap(packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept`
- `template<typename _Tp, typename _Sequence, typename _Compare>`  
`enable_if< __and< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::type swap(priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Res>`  
`void swap(promise< _Res > &__x, promise< _Res > &__y) noexcept`
- `template<typename _Tp, typename _Seq>`  
`enable_if< __is_swappable< _Seq >::value >::type swap(queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`void swap(set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Tp, typename _Seq>`  
`enable_if< __is_swappable< _Seq >::value >::type swap(stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`  
`constexpr enable_if< !__and< __is_swappable< _Elements >... >::value >::type swap(tuple< _Elements... > &__, tuple< _Elements... > &__)=delete`
- `template<typename... _Elements>`  
`constexpr enable_if< __and< __is_swappable< _Elements >... >::value >::type swap(tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Dp>`  
`enable_if< !__is_swappable< _Dp >::value >::type swap(unique_ptr< _Tp, _Dp > &__, unique_ptr< _Tp, _Dp > &__)=delete`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`void swap(unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Alloc>`  
`constexpr void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `void swap (thread &__x, thread &__y) noexcept`
- `template<typename _Mutex>`  
`void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept`
- `template<typename _Tp, typename _Dp>`  
`constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Tp>`  
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `const error_category & system_category () noexcept`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::__value, double >::__type tan (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std:::Tan, _Expr, _Dom >, typename _Dom::value_type > tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > tan (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std:::Tan, _ValArray, _Tp >, _Tp > tan (const valarray< _Tp > &__v)`
- `constexpr float tan (float __x)`
- `constexpr long double tan (long double __x)`
- `template<typename _Tp, typename _Abi, typename _R = _Math_return_type_t< decltype(std::tan(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > tan (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::__value, double >::__type tanh (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std:::Tanh, _Expr, _Dom >, typename _Dom::value_type > tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > tanh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std:::Tanh, _ValArray, _Tp >, _Tp > tanh (const valarray< _Tp > &__v)`
- `constexpr float tanh (float __x)`
- `constexpr long double tanh (long double __x)`

- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tanh(declval<double>())) ), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > tanh (simd< _Tp, _Abi > __x)`
- `void terminate () noexcept`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type tgamma (_Tp __x)`
- `constexpr float tgamma (float __x)`
- `constexpr long double tgamma (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tgamma(declval<double>())) ), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > tgamma (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`void throw_with_nested (_Tp && __t)`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &... > tie (_Elements &... __args) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp * to_address (_Tp * __ptr) noexcept`
- `template<typename _Ptr>`  
`constexpr auto to_address (const _Ptr & __ptr) noexcept`
- `to_chars_result to_chars (char *, char *, bool, int=10)=delete`
- `constexpr to_chars_result to_chars (char * __first, char * __last, char __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, signed char __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, signed int __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, signed long __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, signed long long __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, signed short __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, unsigned char __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, unsigned int __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, unsigned long __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, unsigned long long __value, int __base=10)`
- `constexpr to_chars_result to_chars (char * __first, char * __last, unsigned short __value, int __base=10)`
- `string to_string (double __val)`
- `string to_string (float __val)`
- `string to_string (int __val) noexcept`
- `string to_string (long __val) noexcept`
- `string to_string (long double __val)`
- `string to_string (long long __val)`
- `string to_string (unsigned __val) noexcept`
- `string to_string (unsigned long __val) noexcept`
- `string to_string (unsigned long long __val)`
- `wstring to_wstring (double __val)`
- `wstring to_wstring (float __val)`
- `wstring to_wstring (int __val)`
- `wstring to_wstring (long __val)`
- `wstring to_wstring (long double __val)`
- `wstring to_wstring (long long __val)`
- `wstring to_wstring (unsigned __val)`
- `wstring to_wstring (unsigned long __val)`
- `wstring to_wstring (unsigned long long __val)`

- `template<typename _CharT>`  
`_CharT tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`_CharT toupper (_CharT __c, const locale &__loc)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`constexpr _OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation>`  
`constexpr _OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>`  
`constexpr _OutputIterator transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator transform_exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>`  
`constexpr _OutputIterator transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _Tp transform_reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`  
`constexpr _Tp transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::value, double >::__type trunc (_Tp __x)`
- `constexpr float trunc (float __x)`
- `constexpr long double trunc (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::trunc(declval<double>()))>, _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > trunc (simd< _Tp, _Abi > __x)`
- `template<typename _L1, typename _L2, typename... _L3>`  
`int try_lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)`
- `template<typename... _UTypes>`  
`tuple (_UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename... _UTypes>`  
`tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename _T1, typename _T2>`  
`tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >`
- `template<typename _Alloc, typename... _UTypes>`  
`tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >`
- `template<typename _T1, typename _T2>`  
`tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >`



- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>  
constexpr auto tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result<_Tpls...>::type`
- `bool uncaught_exception () noexcept`
- `void undecare_no_pointers (char *, size_t)`
- `template<typename _Tp>  
_Tp * undecare_reachable (_Tp * __p)`
- `void unexpected ()`
- `template<typename _Tp, typename _Alloc, typename... _Args>  
constexpr _Tp * uninitialized_construct_using_allocator (_Tp * __p, const _Alloc & __a, _Args &&... __args)`
- `template<typename _InputIterator, typename _ForwardIterator>  
_ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>  
_ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator>  
void uninitialized_default_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>  
_ForwardIterator uninitialized_default_construct_n (_ForwardIterator __first, _Size __count)`
- `template<typename _ForwardIterator, typename _Tp>  
void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp>  
_ForwardIterator uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp & __x)`
- `template<typename _InputIterator, typename _ForwardIterator>  
_ForwardIterator uninitialized_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>  
pair< _InputIterator, _ForwardIterator > uninitialized_move_n (_InputIterator __first, _Size __count, _ForwardIterator __result)`
- `template<typename _ForwardIterator>  
void uninitialized_value_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>  
_ForwardIterator uninitialized_value_construct_n (_ForwardIterator __first, _Size __count)`
- `template<typename _Filter>  
constexpr _Filter unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate>  
constexpr _Filter unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator>  
constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>  
constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, typename _OIter>  
constexpr _OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate>  
constexpr _OIter unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _InputIterator, typename _OutputIterator>  
constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>  
constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `ios_base & unitbuf (ios_base & __base)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>  
unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >`



- ```

template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator)
-> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t<
_InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >

• template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename =
_RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash,
_Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash
equal_to< __iter_key_t< _InputIterator > >, _Allocator >

• template<typename _InputIterator, typename _Hash = hash< __iter_key_t<_InputIterator>>, typename _Pred = equal_to< __iter_key_t<
_InputIterator>>, typename _Allocator = allocator< __iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>,
typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_
Allocator>>
unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=_
Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_
val_t< _InputIterator >, _Hash, _Pred, _Allocator >

• template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key
>, equal_to< _Key >, _Allocator >

• template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _
Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >

• template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>,
typename = _RequireAllocator<_Allocator>>
unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Hash,
_Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >

• template<typename _Key, typename _Tp, typename _Hash = hash< _Key >, typename _Pred = equal_to< _Key >, typename _Allocator =
allocator< pair< const _Key, _Tp > >, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>,
typename = _RequireAllocator<_Allocator>>
unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type={},
_Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _
Allocator >

• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require
Allocator<_Allocator>>
unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< __iter_key_t< _
InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_
key_t< _InputIterator > >, _Allocator >

• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require
Allocator<_Allocator>>
unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator) ->
unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t<
_InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >

• template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename =
_RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, _
Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash,
equal_to< __iter_key_t< _InputIterator > >, _Allocator >

• template<typename _InputIterator, typename _Hash = hash< __iter_key_t<_InputIterator>>, typename _Pred = equal_to< __iter_key_t<
_InputIterator>>, typename _Allocator = allocator< __iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>,
typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_
Allocator>>
unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, Hash=

```

- Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_multimap< \_\_iter\_key\_t< \_InputIterator >, \_\_iter\_val\_t< \_InputIterator >, \_Hash, \_Pred, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, \_Allocator) -> unordered\_multimap< \_Key, \_Tp, hash< \_Key >, equal\_to< \_Key >, \_Allocator >
  - template<typename \_Key, typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, unordered\_multimap< int, int >::size\_type, \_Allocator) -> unordered\_multimap< \_Key, \_Tp, hash< \_Key >, equal\_to< \_Key >, \_Allocator >
  - template<typename \_Key, typename \_Tp, typename \_Hash, typename \_Allocator, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, unordered\_multimap< int, int >::size\_type, \_Hash, \_Allocator) -> unordered\_multimap< \_Key, \_Tp, \_Hash, equal\_to< \_Key >, \_Allocator >
  - template<typename \_Key, typename \_Tp, typename \_Hash = hash<\_Key>, typename \_Pred = equal\_to<\_Key>, typename \_Allocator = allocator<pair<const \_Key, \_Tp>>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, unordered\_multimap< int, int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Allocator >
  - template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multiset** (\_InputIterator, \_InputIterator, \_Allocator) -> unordered\_multiset< typename iterator\_traits< \_InputIterator >::value\_type, hash< typename iterator\_traits< \_InputIterator >::value\_type >, equal\_to< typename iterator\_traits< \_InputIterator >::value\_type >, \_Allocator >
  - template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multiset** (\_InputIterator, \_InputIterator, unordered\_multiset< int >::size\_type, \_Allocator) -> unordered\_multiset< typename iterator\_traits< \_InputIterator >::value\_type, hash< typename iterator\_traits< \_InputIterator >::value\_type >, equal\_to< typename iterator\_traits< \_InputIterator >::value\_type >, \_Allocator >
  - template<typename \_InputIterator, typename \_Hash, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multiset** (\_InputIterator, \_InputIterator, unordered\_multiset< int >::size\_type, \_Hash, \_Allocator) -> unordered\_multiset< typename iterator\_traits< \_InputIterator >::value\_type, \_Hash, equal\_to< typename iterator\_traits< \_InputIterator >::value\_type >, \_Allocator >
  - template<typename \_InputIterator, typename \_Hash = hash<typename iterator\_traits<\_InputIterator>::value\_type>, typename \_Pred = equal\_to<typename iterator\_traits<\_InputIterator>::value\_type>, typename \_Allocator = allocator<typename iterator\_traits<\_InputIterator>::value\_type>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multiset** (\_InputIterator, \_InputIterator, unordered\_multiset< int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_multiset< typename iterator\_traits< \_InputIterator >::value\_type, \_Hash, \_Pred, \_Allocator >
  - template<typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multiset** (initializer\_list< \_Tp >, \_Allocator) -> unordered\_multiset< \_Tp, hash< \_Tp >, equal\_to< \_Tp >, \_Allocator >
  - template<typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multiset** (initializer\_list< \_Tp >, unordered\_multiset< int >::size\_type, \_Allocator) -> unordered\_multiset< \_Tp, hash< \_Tp >, equal\_to< \_Tp >, \_Allocator >
  - template<typename \_Tp, typename \_Hash, typename \_Allocator, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multiset** (initializer\_list< \_Tp >, unordered\_multiset< int >::size\_type, \_Hash, \_Allocator) -> unordered\_multiset< \_Tp, \_Hash, equal\_to< \_Tp >, \_Allocator >
  - template<typename \_Tp, typename \_Hash = hash<\_Tp>, typename \_Pred = equal\_to<\_Tp>, typename \_Allocator = allocator<\_Tp>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multiset** (initializer\_list< \_Tp >, unordered\_multiset< int >::size\_type, \_Hash, \_Allocator) -> unordered\_multiset< \_Tp, \_Hash, equal\_to< \_Tp >, \_Allocator >

```

unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >
• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
• template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
• template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
• template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
• template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
• template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >
• template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >
• template<typename _Filter, typename _Tp>
constexpr _Filter upper_bound (_Filter, _Filter, const _Tp &)
• template<typename _Filter, typename _Tp, typename _Compare>
constexpr _Filter upper_bound (_Filter, _Filter, const _Tp &, _Compare)
• template<typename _ForwardIterator, typename _Tp>
constexpr _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
• template<typename _ForwardIterator, typename _Tp, typename _Compare>
constexpr _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
• ios_base & uppercase (ios_base &__base)
• template<typename _Facet>
const _Facet & use_facet (const locale &__loc)
• template const codecvt< char, char, mbstate_t > & use_facet< codecvt< char, char, mbstate_t > > (const locale &)

```

- template const `codecvt`< wchar\_t, char, mbstate\_t > & `use_facet`< `codecvt`< wchar\_t, char, mbstate\_t > > (const `locale` &)
- template<\_Std\_pair \_Tp, typename \_Alloc>  
constexpr auto `uses_allocator_construction_args` (const \_Alloc &) noexcept
- template<\_Std\_pair \_Tp, typename \_Alloc, typename \_Up, typename \_Vp>  
constexpr auto `uses_allocator_construction_args` (const \_Alloc &, \_Up &&, \_Vp &&) noexcept
- template<\_Std\_pair \_Tp, typename \_Alloc, typename \_Up, typename \_Vp>  
constexpr auto `uses_allocator_construction_args` (const \_Alloc &, const `pair`< \_Up, \_Vp > &) noexcept
- template<\_Std\_pair \_Tp, typename \_Alloc, typename \_Up, typename \_Vp>  
constexpr auto `uses_allocator_construction_args` (const \_Alloc &, `pair`< \_Up, \_Vp > &&) noexcept
- template<typename \_Tp, typename \_Alloc, typename... \_Args>  
requires (! \_Std\_pair<\_Tp>)  
constexpr auto `uses_allocator_construction_args` (const \_Alloc & \_\_a, \_Args &&... \_\_args) noexcept
- template<\_Std\_pair \_Tp, typename \_Alloc, typename \_Tuple1, typename \_Tuple2>  
constexpr auto `uses_allocator_construction_args` (const \_Alloc & \_\_a, `piecewise_construct_t`, \_Tuple1 && \_\_x, \_Tuple2 && \_\_y) noexcept
- template<typename \_Tp, size\_t \_Nm>  
`valarray` (const \_Tp(&)[\_Nm], size\_t) -> `valarray`< \_Tp >
- template<typename \_InputIterator, typename \_ValT = typename iterator\_traits<\_InputIterator>::value\_type, typename \_Allocator = allocator<\_ValT>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>  
`vector` (\_InputIterator, \_InputIterator, \_Allocator=\_Allocator()) -> `vector`< \_ValT, \_Allocator >
- wchar\_t \* `wcschr` (wchar\_t \* \_\_p, wchar\_t \_\_c)
- wchar\_t \* `wcspbrk` (wchar\_t \* \_\_s1, const wchar\_t \* \_\_s2)
- wchar\_t \* `wcsrchr` (wchar\_t \* \_\_p, wchar\_t \_\_c)
- wchar\_t \* `wcsstr` (wchar\_t \* \_\_s1, const wchar\_t \* \_\_s2)
- template<typename \_Tp>  
`weak_ptr` (`shared_ptr`< \_Tp >) -> `weak_ptr`< \_Tp >
- wchar\_t \* `wmemchr` (wchar\_t \* \_\_p, wchar\_t \_\_c, size\_t \_\_n)
- template<typename \_CharT, typename \_Traits>  
`basic_istream`< \_CharT, \_Traits > & `ws` (`basic_istream`< \_CharT, \_Traits > & \_\_is)
- template<size\_t \_Nb>  
constexpr `bitset`< \_Nb > `operator&` (const `bitset`< \_Nb > & \_\_x, const `bitset`< \_Nb > & \_\_y) noexcept
- template<size\_t \_Nb>  
constexpr `bitset`< \_Nb > `operator|` (const `bitset`< \_Nb > & \_\_x, const `bitset`< \_Nb > & \_\_y) noexcept
- template<size\_t \_Nb>  
constexpr `bitset`< \_Nb > `operator^` (const `bitset`< \_Nb > & \_\_x, const `bitset`< \_Nb > & \_\_y) noexcept
- template<class \_CharT, class \_Traits, size\_t \_Nb>  
`std::basic_istream`< \_CharT, \_Traits > & `operator>>` (`std::basic_istream`< \_CharT, \_Traits > & \_\_is, `bitset`< \_Nb > & \_\_x)
- template<class \_CharT, class \_Traits, size\_t \_Nb>  
`std::basic_ostream`< \_CharT, \_Traits > & `operator<<` (`std::basic_ostream`< \_CharT, \_Traits > & \_\_os, const `bitset`< \_Nb > & \_\_x)
- template<typename \_Tp>  
constexpr `complex`< \_Tp > `operator+` (const `complex`< \_Tp > & \_\_x, const `complex`< \_Tp > & \_\_y)
- template<typename \_Tp>  
constexpr `complex`< \_Tp > `operator+` (const `complex`< \_Tp > & \_\_x, const \_Tp & \_\_y)

- `template<typename _Tp>`  
`constexpr complex< _Tp > operator+ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator- (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator* (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr bool operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits>`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits>`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`
- `template<typename _CharT, typename _Traits, size_t _Num>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT(&__s)[_Num])`
- `template<class _Traits, size_t _Num>`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char(&__s)[_Num])`
- `template<class _Traits, size_t _Num>`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char(&__s)[_Num])`

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, wchar_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, char16_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, char32_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, char16_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, char32_t)=delete`
  
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, const wchar_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, const char16_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, const char32_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, const char16_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, const char32_t *)=delete`

### Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`



- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match ( _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits>`  
`bool regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const`  
`basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch←`  
`_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type,`  
`_Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Ch_type, class _Rx_traits>`  
`bool regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type`  
`__f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex<`  
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search ( _Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex<`  
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search ( _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits>`  
`bool regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const`  
`basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits>`  
`bool regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type`  
`__f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex<`  
`_Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch←`  
`_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type,`  
`_Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`_Out_iter regex_replace ( _Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, ←`  
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type`  
`__flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>`  
`_Out_iter regex_replace ( _Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, ←`  
`_Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>`  
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__←`  
`s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const`

- `basic_regex<_Ch_type, _Rx_traits> &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __↵  
flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>  
basic_string<_Ch_type> regex_replace (const _Ch_type *__s, const basic_regex<_Ch_type, _Rx_↵  
traits> &__e, const basic_string<_Ch_type, _St, _Sa> &__fmt, regex_constants::match_flag_type __↵  
flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type>  
basic_string<_Ch_type> regex_replace (const _Ch_type *__s, const basic_regex<_Ch_type, _Rx_traits>  
&__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Tp, _Lock_policy _Lp>  
bool atomic_is_lock_free (const __shared_ptr<_Tp, _Lp> *)`
- `template<typename _Tp>  
bool atomic_is_lock_free (const shared_ptr<_Tp> *__p)`
- `template<typename _Tp>  
shared_ptr<_Tp> atomic_load_explicit (const shared_ptr<_Tp> *__p, memory_order)`
- `template<typename _Tp>  
shared_ptr<_Tp> atomic_load (const shared_ptr<_Tp> *__p)`
- `template<typename _Tp, _Lock_policy _Lp>  
__shared_ptr<_Tp, _Lp> atomic_load_explicit (const __shared_ptr<_Tp, _Lp> *__p, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>  
__shared_ptr<_Tp, _Lp> atomic_load (const __shared_ptr<_Tp, _Lp> *__p)`
- `template<typename _Tp>  
void atomic_store_explicit (shared_ptr<_Tp> *__p, shared_ptr<_Tp> __r, memory_order)`
- `template<typename _Tp>  
void atomic_store (shared_ptr<_Tp> *__p, shared_ptr<_Tp> __r)`
- `template<typename _Tp, _Lock_policy _Lp>  
void atomic_store_explicit (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp, _Lp> __r, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>  
void atomic_store (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp, _Lp> __r)`
- `template<typename _Tp>  
shared_ptr<_Tp> atomic_exchange_explicit (shared_ptr<_Tp> *__p, shared_ptr<_Tp> __↵  
r, memory_order)`
- `template<typename _Tp>  
shared_ptr<_Tp> atomic_exchange (shared_ptr<_Tp> *__p, shared_ptr<_Tp> __r)`
- `template<typename _Tp, _Lock_policy _Lp>  
__shared_ptr<_Tp, _Lp> atomic_exchange_explicit (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp,  
_Lp> __r, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>  
__shared_ptr<_Tp, _Lp> atomic_exchange (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp, _Lp> __r)`
- `template<typename _Tp>  
bool atomic_compare_exchange_strong_explicit (shared_ptr<_Tp> *__p, shared_ptr<_Tp> *__↵  
v, shared_ptr<_Tp> __w, memory_order, memory_order)`



- `template<typename _Tp>`  
`bool atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp>`  
`bool atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp>`  
`bool atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & get (pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && get (pair< _Tp1, _Tp2 > &&__in) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & get (const pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && get (const pair< _Tp1, _Tp2 > &&__in) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp & get (pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp & get (const pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp && get (pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp && get (const pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp & get (pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp & get (const pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp && get (pair< _Up, _Tp > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp && get (const pair< _Up, _Tp > &&__p) noexcept`

## Variables

- `template<typename _ValT, typename _Tp>`  
`constexpr bool __can_use_memchr_for_find`
- `const _Lock_policy __default_lock_policy`
- `template<typename _Tp>`  
`constexpr auto __denorm_min_v`
- `template<typename _Tp>`  
`constexpr auto __digits10_v`
- `template<typename _Tp>`  
`constexpr auto __digits_v`
- `template<typename _Tp>`  
`constexpr auto __epsilon_v`
- `template<typename _Tp>`  
`constexpr auto __finite_max_v`
- `template<typename _Tp>`  
`constexpr auto __finite_min_v`
- `template<template< typename > class _Trait, typename _Tp>`  
`constexpr bool __has_iec559_behavior_v`
- `template<typename _Tp>`  
`constexpr bool __has_iec559_storage_format_v`
- `template<typename _Tp>`  
`constexpr auto __infinity_v`
- `static ios_base::Init __ioint`
- `template<typename _Tp, typename = void>`  
`constexpr bool __is_hash_enabled_for`
- `template<typename _Tp>`  
`constexpr bool __is_hash_enabled_for< _Tp, __void_t< decltype(hash< _Tp >())(declval< _Tp >())>> >`
- `template<typename>`  
`constexpr bool __is_in_place_index_v`
- `template<size_t _Nm>`  
`constexpr bool __is_in_place_index_v< in_place_index_t< _Nm > >`
- `template<typename>`  
`constexpr bool __is_in_place_type_v`
- `template<typename _Tp>`  
`constexpr bool __is_in_place_type_v< in_place_type_t< _Tp > >`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool __is_nothrow_new_constructible`
- `template<typename _Tp>`  
`constexpr bool __is_pair`
- `template<typename _Tp, typename _Up>`  
`constexpr bool __is_pair< pair< _Tp, _Up > >`
- `template<typename _Tp>`  
`constexpr auto __max_digits10_v`
- `template<typename _Tp>`  
`constexpr auto __max_exponent10_v`
- `template<typename _Tp>`  
`constexpr auto __max_exponent_v`
- `template<typename _Tp>`  
`constexpr auto __min_exponent10_v`
- `template<typename _Tp>`  
`constexpr auto __min_exponent_v`

- `template<typename _Tp>`  
`constexpr auto __norm_min_v`
- `template<typename _Tp>`  
`constexpr auto __quiet_NaN_v`
- `template<typename _Tp>`  
`constexpr auto __radix_v`
- `template<typename _Tp>`  
`constexpr auto __reciprocal_overflow_threshold_v`
- `template<typename _Tp>`  
`constexpr auto __round_error_v`
- `template<typename _Tp>`  
`constexpr auto __signaling_NaN_v`
- `template<typename _CharT>`  
`locale::id __timepunct<_CharT>::id`
- `template<typename _CharT>`  
`const _CharT * __timepunct_cache<_CharT>::S_timezones [14]`
- `template<> const char * __timepunct_cache<char>::S_timezones [14]`
- `template<> const wchar_t * __timepunct_cache<wchar_t>::S_timezones [14]`
- `template<template< typename > class _Trait, typename _Tp>`  
`constexpr bool __value_exists_v`
- `constexpr adopt\_lock\_t adopt_lock`
- `template<size_t _Len, typename... _Types>`  
`const size_t aligned_union<_Len, _Types...>::alignment_value`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`const basic\_string<\_CharT, \_Traits, \_Alloc>::size\_type basic_string<_CharT, _Traits, _Alloc>::npos`
- `template<typename _InternT, typename _ExternT, typename _StateT>`  
`locale::id codecvt<_InternT, _ExternT, _StateT>::id`
- `template<typename _InternT, typename _ExternT>`  
`locale::id codecvt<_InternT, _ExternT, encoding_state>::id`
- `template<typename _CharT>`  
`locale::id collate<_CharT>::id`
- `constexpr __compare::Partial_fallback compare_partial_order_fallback`
- `constexpr __compare::Strong_fallback compare_strong_order_fallback`
- `constexpr __compare::Weak_fallback compare_weak_order_fallback`
- `template<typename _CharT>`  
`locale::id ctype<_CharT>::id`
- `constexpr default\_sentinel\_t default_sentinel`
- `constexpr defer\_lock\_t defer_lock`
- `constexpr destroying\_delete\_t destroying_delete`
- `template<typename _Iterator1, typename _Iterator2>`  
`constexpr bool disable_sized_sentinel_for<move_iterator<_Iterator1>, move_iterator<_Iterator2>>`
- `template<typename _Iterator1, typename _Iterator2>`  
`constexpr bool disable_sized_sentinel_for<reverse_iterator<_Iterator1>, reverse_iterator<_Iterator2>>`  
`>`
- `constexpr _Swallow_assign ignore`
- `constexpr in_place_t in_place`
- `template<size_t _Idx>`  
`constexpr in_place_index_t<_Idx> in_place_index`
- `template<typename _Tp>`  
`constexpr in_place_type_t<_Tp> in_place_type`
- `template<typename _Tp>`  
`constexpr bool is_bind_expression_v`

- `template<typename _Tp>`  
`constexpr bool is_error_code_enum_v`
- `template<typename _Tp>`  
`constexpr bool is_error_condition_enum_v`
- `template<typename _Tp>`  
`constexpr int is_placeholder_v`
- `constexpr memory\_order memory_order_acq_rel`
- `constexpr memory\_order memory_order_acquire`
- `constexpr memory\_order memory_order_consume`
- `constexpr memory\_order memory_order_relaxed`
- `constexpr memory\_order memory_order_release`
- `constexpr memory\_order memory_order_seq_cst`
- `template<typename _CharT>`  
`locale::id messages< _CharT >::id`
- `template<typename _CharT, typename _InIter>`  
`locale::id money_get< _CharT, _InIter >::id`
- `template<typename _CharT, typename _OutIter>`  
`locale::id money_put< _CharT, _OutIter >::id`
- `template<typename _CharT, bool _Intl>`  
`locale::id moneypunct< _CharT, _Intl >::id`
- `template<typename _CharT, bool _Intl>`  
`const bool moneypunct< _CharT, _Intl >::intl`
- `template<typename _CharT, bool _Intl>`  
`const bool moneypunct_byname< _CharT, _Intl >::intl`
- `const nothrow_t nothrow`
- `template<typename _CharT, typename _InIter>`  
`locale::id num_get< _CharT, _InIter >::id`
- `template<typename _CharT, typename _OutIter>`  
`locale::id num_put< _CharT, _OutIter >::id`
- `template<typename _CharT>`  
`locale::id numpunct< _CharT >::id`
- `constexpr __compare::Partial_order partial_order`
- `constexpr piecewise\_construct\_t piecewise_construct`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_greater_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_greater_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_less_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_less_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_not_equal_v`
- `constexpr __compare::Strong_order strong_order`
- `template<typename _CharT, typename _InIter>`  
`locale::id time_get< _CharT, _InIter >::id`
- `template<typename _CharT, typename _OutIter>`  
`locale::id time_put< _CharT, _OutIter >::id`
- `constexpr try\_to\_lock\_t try_to_lock`

- `template<typename _Tp>`  
`constexpr size_t tuple_size_v`
- `template<typename _Tp, size_t _Nm>`  
`constexpr size_t tuple_size_v< array< _Tp, _Nm > >`
- `template<typename _Tp, size_t _Nm>`  
`constexpr size_t tuple_size_v< const array< _Tp, _Nm > >`
- `template<typename _Tp1, typename _Tp2>`  
`constexpr size_t tuple_size_v< const pair< _Tp1, _Tp2 > >`
- `template<typename... _Types>`  
`constexpr size_t tuple_size_v< const tuple< _Types... > >`
- `template<typename _Tp1, typename _Tp2>`  
`constexpr size_t tuple_size_v< pair< _Tp1, _Tp2 > >`
- `template<typename... _Types>`  
`constexpr size_t tuple_size_v< tuple< _Types... > >`
- `constexpr __compare::__Weak_order weak_order`

### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html> and the [I/O forward declarations](#)

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- [istream cin](#)
- [ostream cout](#)
- [ostream cerr](#)
- [ostream clog](#)
- [wistream wcin](#)
- [wostream wcout](#)
- [wostream wcerr](#)
- [wostream wclog](#)
- `template<typename... _Tp>`  
`using common_reference_t`
- `template<typename _Tp, typename _Up>`  
`constexpr bool is_layout_compatible_v`
- `template<typename _Base, typename _Derived>`  
`constexpr bool is_pointer_interconvertible_base_of_v`
- `template<typename _S1, typename _S2, typename _M1, typename _M2>`  
`constexpr bool is_corresponding_member (_M1 _S1::*__m1, _M2 _S2::*__m2) noexcept`
- `template<typename _Tp, typename _Mem>`  
`constexpr bool is_pointer_interconvertible_with_class (_Mem _Tp::*__mp) noexcept`

#### 4.10.1 Detailed Description

ISO C++ entities toplevel namespace is `std`.

#### 4.10.2 Typedef Documentation

##### `__ptr_rebind`

```
template<typename _Ptr, typename _Tp>
using std::__ptr_rebind
```

Convenience alias for rebinding pointers.

**\_\_umap\_traits**

```
template<bool _Cache>
using std::__umap_traits
Base types for unordered_map.
```

**\_\_ummap\_traits**

```
template<bool _Cache>
using std::__ummap_traits
Base types for unordered_multimap.
```

**\_\_umset\_traits**

```
template<bool _Cache>
using std::__umset_traits
Base types for unordered_multiset.
```

**\_\_uset\_traits**

```
template<bool _Cache>
using std::__uset_traits
Base types for unordered_set.
```

**compare\_three\_way\_result\_t**

```
template<typename _Tp, typename _Up = _Tp>
using std::compare_three_way_result_t
[cmp.result], result of three-way comparison
```

**index\_sequence**

```
template<size_t... _Idx>
using std::index_sequence
Alias template index_sequence.
```

**index\_sequence\_for**

```
template<typename... _Types>
using std::index_sequence_for
Alias template index_sequence_for.
```

**make\_index\_sequence**

```
template<size_t _Num>
using std::make_index_sequence
Alias template make_index_sequence.
```

**make\_integer\_sequence**

```
template<typename _Tp, _Tp _Num>
using std::make_integer_sequence
Alias template make_integer_sequence.
```

**new\_handler**

```
typedef void(* std::new_handler) ()
```

If you write your own error handler to be called by `new`, it must be of this type.

**streamoff**

```
typedef long long std::streamoff
```

Type used by `fpos`, `char_traits<char>`, and `char_traits<wchar_t>`.

In clauses 21.1.3.1 and 27.4.1 `streamoff` is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, `streamoff` was `typedef long`.

**streampos**

```
typedef fpos<mbstate_t> std::streampos
```

File position for char streams.

**streamsize**

```
typedef ptrdiff_t std::streamsize
```

Integral type for I/O operation counts and buffer sizes.

**u16streampos**

```
typedef fpos<mbstate_t> std::u16streampos
```

File position for `char16_t` streams.

**u32streampos**

```
typedef fpos<mbstate_t> std::u32streampos
```

File position for `char32_t` streams.

**wstreampos**

```
typedef fpos<mbstate_t> std::wstreampos
```

File position for `wchar_t` streams.

**4.10.3 Enumeration Type Documentation****chars\_format**

```
enum class std::chars_format [strong]
```

floating-point format for primitive numerical conversion

**float\_denorm\_style**

```
enum std::float_denorm_style
```

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the `std::numeric_limits` class.

**Enumerator**

|                                   |                                                                        |
|-----------------------------------|------------------------------------------------------------------------|
| <code>denorm_indeterminate</code> | Indeterminate at compile time whether denormalized values are allowed. |
| <code>denorm_absent</code>        | The type does not allow denormalized values.                           |
| <code>denorm_present</code>       | The type allows denormalized values.                                   |

**float\_round\_style**

```
enum std::float_round_style
```

Describes the rounding style for floating-point types.

This is used in the `std::numeric_limits` class.

**Enumerator**

|                                        |                                     |
|----------------------------------------|-------------------------------------|
| <code>round_indeterminate</code>       | Intermediate.                       |
| <code>round_toward_zero</code>         | To zero.                            |
| <code>round_to_nearest</code>          | To the nearest representable value. |
| <code>round_toward_infinity</code>     | To infinity.                        |
| <code>round_toward_neg_infinity</code> | To negative infinity.               |

**io\_errc**

```
enum class std::io_errc [strong]
```

I/O error code.

**4.10.4 Function Documentation****`__find_if_not()`**

```
template<typename _InputIterator, typename _Predicate>
_InputIterator std::__find_if_not (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred) [inline], [constexpr]
```

Provided for `stable_partition` to use.

Referenced by [find\\_if\\_not\(\)](#).

**`__find_if_not_n()`**

```
template<typename _InputIterator, typename _Predicate, typename _Distance>
_InputIterator std::__find_if_not_n (
    _InputIterator __first,
    _Distance & __len,
    _Predicate __pred) [constexpr]
```

Like `find_if_not()`, but uses and updates a count of the remaining range length instead of comparing against an end iterator.

Referenced by [\\_\\_stable\\_partition\\_adaptive\(\)](#).

**`__gcd()`**

```
template<typename _EuclideanRingElement>
_EuclideanRingElement std::__gcd (
    _EuclideanRingElement __m,
    _EuclideanRingElement __n) [constexpr]
```

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.



**\_\_gen\_two\_uniform\_ints()**

```
template<typename _IntType, typename _UniformRandomBitGenerator>
pair< _IntType, _IntType > std::__gen_two_uniform_ints (
    _IntType __b0,
    _IntType __b1,
    _UniformRandomBitGenerator && __g)
```

Generate two uniformly distributed integers using a single distribution invocation.

**Parameters**

|                   |                                         |
|-------------------|-----------------------------------------|
| <code>__b0</code> | The upper bound for the first integer.  |
| <code>__b1</code> | The upper bound for the second integer. |
| <code>__g</code>  | A UniformRandomBitGenerator.            |

**Returns**

A pair (i, j) with i and j uniformly distributed over [0, \_\_b0) and [0, \_\_b1), respectively.

Requires: `__b0 * __b1 <= __g.max() - __g.min()`.

Using `uniform_int_distribution` with a range that is very small relative to the range of the generator ends up wasting potentially expensively generated randomness, since `uniform_int_distribution` does not store leftover randomness between invocations.

If we know we want two integers in ranges that are sufficiently small, we can compose the ranges, use a single distribution invocation, and significantly reduce the waste.

References [make\\_pair\(\)](#).

Referenced by [\\_\\_sample\(\)](#), and [shuffle\(\)](#).

**\_\_inplace\_stable\_sort()**

```
template<typename _RandomAccessIterator, typename _Compare>
void std::__inplace_stable_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp)
```

This is a helper function for the stable sorting routines.

References [\\_\\_inplace\\_stable\\_sort\(\)](#), and [\\_\\_merge\\_without\\_buffer\(\)](#).

Referenced by [\\_\\_inplace\\_stable\\_sort\(\)](#).

**\_\_lg()**

```
template<typename _Tp>
_Tp std::__lg (
    _Tp __n) [inline], [constexpr]
```

This is a helper function for the sort routines and for `random.tcc`.

Referenced by [nth\\_element\(\)](#), [nth\\_element\(\)](#), [std::independent\\_bits\\_engine<\\_RandomNumberEngine, \\_\\_w, \\_UIntType >::operator\(\)\(\)](#), and [std::linear\\_congruential\\_engine<uint\\_fast32\\_t, 16807UL, 0UL, 2147483647UL >::seed\(\)](#).

**\_\_merge\_adaptive()**

```
template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _↵
Compare>
void std::__merge_adaptive (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
```

```

        _BidirectionalIterator __last,
        _Distance __len1,
        _Distance __len2,
        _Pointer __buffer,
        _Compare __comp)

```

This is a helper function for the merge routines.

References [\\_\\_move\\_merge\\_adaptive\(\)](#), and [\\_\\_move\\_merge\\_adaptive\\_backward\(\)](#).

### **`__merge_without_buffer()`**

```

template<typename _BidirectionalIterator, typename _Distance, typename _Compare>
void std::__merge_without_buffer (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last,
    _Distance __len1,
    _Distance __len2,
    _Compare __comp)

```

This is a helper function for the merge routines.

References [\\_\\_merge\\_without\\_buffer\(\)](#), [advance\(\)](#), and [distance\(\)](#).

Referenced by [\\_\\_inplace\\_stable\\_sort\(\)](#), and [\\_\\_merge\\_without\\_buffer\(\)](#).

### **`__move_median_to_first()`**

```

template<typename _Iterator, typename _Compare>
void std::__move_median_to_first (
    _Iterator __result,
    _Iterator __a,
    _Iterator __b,
    _Iterator __c,
    _Compare __comp) [constexpr]

```

Swaps the median value of `*__a`, `*__b` and `*__c` under `__comp` to `*__result`.

### **`__move_merge()`**

```

template<typename _InputIterator, typename _OutputIterator, typename _Compare>
_OutputIterator std::__move_merge (
    _InputIterator __first1,
    _InputIterator __last1,
    _InputIterator __first2,
    _InputIterator __last2,
    _OutputIterator __result,
    _Compare __comp)

```

This is a helper function for the `__merge_sort_loop` routines.

### **`__move_merge_adaptive()`**

```

template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
void std::__move_merge_adaptive (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,

```

```

    __OutputIterator __result,
    __Compare __comp)

```

This is a helper function for the `__merge_adaptive` routines.

Referenced by [\\_\\_merge\\_adaptive\(\)](#).

### **`__move_merge_adaptive_backward()`**

```

template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare>
void std::__move_merge_adaptive_backward (
    _BidirectionalIterator1 __first1,
    _BidirectionalIterator1 __last1,
    _BidirectionalIterator2 __first2,
    _BidirectionalIterator2 __last2,
    _BidirectionalIterator3 __result,
    __Compare __comp)

```

This is a helper function for the `__merge_adaptive` routines.

Referenced by [\\_\\_merge\\_adaptive\(\)](#).

### **`__partition()` [1/2]**

```

template<typename _BidirectionalIterator, typename _Predicate>
_BidirectionalIterator std::__partition (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _Predicate __pred,
    bidirectional\_iterator\_tag ) [constexpr]

```

This is a helper function...

### **`__partition()` [2/2]**

```

template<typename _ForwardIterator, typename _Predicate>
_FowardIterator std::__partition (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred,
    forward\_iterator\_tag ) [constexpr]

```

This is a helper function...

Referenced by [partition\(\)](#).

### **`__reverse()` [1/2]**

```

template<typename _BidirectionalIterator>
void std::__reverse (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    bidirectional\_iterator\_tag ) [constexpr]

```

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for bidirectional iterators.

Referenced by [\\_\\_rotate\(\)](#), and [reverse\(\)](#).

### **`__reverse()` [2/2]**

```

template<typename _RandomAccessIterator>
void std::__reverse (
    _RandomAccessIterator __first,

```

```

    _RandomAccessIterator __last,
    random_access_iterator_tag ) [constexpr]

```

This is an uglified reverse([\\_BidirectionalIterator](#), [\\_BidirectionalIterator](#)) overloaded for random access iterators.

#### **\_\_rotate()** [1/3]

```

template<typename _BidirectionalIterator>
_BidirectionalIterator std::__rotate (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last,
    bidirectional_iterator_tag ) [constexpr]

```

This is a helper function for the rotate algorithm.

References [\\_\\_reverse\(\)](#).

#### **\_\_rotate()** [2/3]

```

template<typename _ForwardIterator>
_FowardIterator std::__rotate (
    _ForwardIterator __first,
    _ForwardIterator __middle,
    _ForwardIterator __last,
    forward_iterator_tag ) [constexpr]

```

This is a helper function for the rotate algorithm.

Referenced by [rotate\(\)](#).

#### **\_\_rotate()** [3/3]

```

template<typename _RandomAccessIterator>
_RandomAccessIterator std::__rotate (
    _RandomAccessIterator __first,
    _RandomAccessIterator __middle,
    _RandomAccessIterator __last,
    random_access_iterator_tag ) [constexpr]

```

This is a helper function for the rotate algorithm.

#### **\_\_rotate\_adaptive()**

```

template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance>
_BidirectionalIterator1 std::__rotate_adaptive (
    _BidirectionalIterator1 __first,
    _BidirectionalIterator1 __middle,
    _BidirectionalIterator1 __last,
    _Distance __len1,
    _Distance __len2,
    _BidirectionalIterator2 __buffer,
    _Distance __buffer_size)

```

This is a helper function for the merge routines.

#### **\_\_sample()** [1/2]

```

template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size,
typename _UniformRandomBitGenerator>
_OutputIterator std::__sample (
    _ForwardIterator __first,

```

```

    _ForwardIterator __last,
    forward_iterator_tag ,
    _OutputIterator __out,
    _Cat ,
    _Size __n,
    _UniformRandomBitGenerator && __g)

```

Selection sampling algorithm.

References [\\_\\_gen\\_two\\_uniform\\_ints\(\)](#), [distance\(\)](#), [std::pair<\\_T1, \\_T2>::first](#), [min\(\)](#), and [std::pair<\\_T1, \\_T2>::second](#).

### **\_\_sample()** [2/2]

```

template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _↵
UniformRandomBitGenerator>
_RandomAccessIterator std::__sample (
    _InputIterator __first,
    _InputIterator __last,
    input_iterator_tag ,
    _RandomAccessIterator __out,
    random_access_iterator_tag ,
    _Size __n,
    _UniformRandomBitGenerator && __g)

```

Reservoir sampling algorithm.

### **\_\_search\_n\_aux()** [1/2]

```

template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>
_FowardIterator std::__search_n_aux (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Integer __count,
    _UnaryPredicate __unary_pred,
    std::forward_iterator_tag ) [constexpr]

```

This is an helper function for `search_n` overloaded for forward iterators.

### **\_\_search\_n\_aux()** [2/2]

```

template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate>
_RandomAccessIter std::__search_n_aux (
    _RandomAccessIter __first,
    _RandomAccessIter __last,
    _Integer __count,
    _UnaryPredicate __unary_pred,
    std::random_access_iterator_tag ) [constexpr]

```

This is an helper function for `search_n` overloaded for random access iterators.

### **\_\_stable\_partition\_adaptive()**

```

template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>
_FowardIterator std::__stable_partition_adaptive (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred,
    _Distance __len,
    _Pointer __buffer,
    _Distance __buffer_size)

```

This is a helper function... Requires `__first != __last` and `!__pred(__first)` and `__len == distance(__first, __last)`.

`!__pred(__first)` allows us to guarantee that we don't move-assign an element onto itself.

References [\\_\\_find\\_if\\_not\\_n\(\)](#), [\\_\\_stable\\_partition\\_adaptive\(\)](#), and [advance\(\)](#).

Referenced by [\\_\\_stable\\_partition\\_adaptive\(\)](#).

### **`_Construct()`**

```
template<typename _Tp, typename... _Args>
void std::_Construct (
    _Tp * __p,
    _Args &&... __args) [inline], [constexpr]
```

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

References [forward\(\)](#).

Referenced by [std::allocator\\_traits< \\_OuterAlloc >::construct\(\)](#), [std::allocator\\_traits< allocator< \\_Tp > >::construct\(\)](#), and [std::allocator\\_traits< allocator< void > >::construct\(\)](#).

### **`_Destroy()` [1/2]**

```
template<typename _ForwardIterator>
void std::_Destroy (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [constexpr]
```

Destroy a range of objects. If the `value_type` of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

References [\\_Destroy\(\)](#), and [addressof\(\)](#).

Referenced by [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >::~~vector\(\)](#), [\\_Destroy\(\)](#), [\\_Destroy\\_n\(\)](#), [std::deque< \\_Tp, \\_Alloc >::M\\_range\\_initialize\(\)](#), [std::allocator\\_traits< \\_OuterAlloc >::destroy\(\)](#), [std::allocator\\_traits< allocator< void > >::operator=\(\)](#), and [std::vector< \\_Tp, \\_Alloc >::reserve\(\)](#).

### **`_Destroy()` [2/2]**

```
template<typename _Tp>
void std::_Destroy (
    _Tp * __pointer) [inline], [constexpr]
```

Destroy the object pointed to by a pointer type.

### **`_Destroy_n()`**

```
template<typename _ForwardIterator, typename _Size>
void std::_Destroy_n (
    _ForwardIterator __first,
    _Size __count) [inline], [constexpr]
```

Destroy a range of objects. If the `value_type` of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

References [\\_Destroy\(\)](#), [addressof\(\)](#), and [advance\(\)](#).

### **`acosh()`**

```
float std::acosh (
    float __x) [constexpr]
```

Additional overloads.

### **`advance()`**

```
template<typename _InputIterator, typename _Distance>
```

```
void std::advance (
    _InputIterator & __i,
    _Distance __n) [inline], [constexpr]
```

A generalization of pointer arithmetic.

#### Parameters

|                     |                                          |
|---------------------|------------------------------------------|
| $\leftarrow$<br>__i | An input iterator.                       |
| $\leftarrow$<br>__n | The <i>delta</i> by which to change __i. |

#### Returns

Nothing.

This increments *i* by *n*. For bidirectional and random access iterators, *\_\_n* may be negative, in which case *\_\_i* is decremented.

For random access iterators, this uses their + and – operations and are constant time. For other iterator classes they are linear time.

References [\\_\\_iterator\\_category\(\)](#).

Referenced by [\\_\\_merge\\_without\\_buffer\(\)](#), [\\_\\_stable\\_partition\\_adaptive\(\)](#), [\\_Destroy\\_n\(\)](#), [std::deque<\\_Tp, \\_Alloc>::\\_M\\_range\\_initialize\(\)](#), [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer< node, leaf, head, inode, const\\_iterator, iterator, \\_Alloc >::get\\_child\(\)](#), [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter< node, leaf, head, inode, const\\_iterator, iterator, \\_Alloc >::get\\_child\(\)](#), and [partition\\_point\(\)](#).

#### **begin()** [1/4]

```
template<typename _Container>
auto std::begin (
    _Container & __cont) -> decltype(__cont.begin()) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the container.

#### Parameters

|        |            |
|--------|------------|
| __cont | Container. |
|--------|------------|

#### **begin()** [2/4]

```
template<typename _Tp, size_t _Nm>
_Tp * std::begin (
    _Tp(&) __arr[_Nm]) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the array.

#### Parameters

|       |        |
|-------|--------|
| __arr | Array. |
|-------|--------|

#### **begin()** [3/4]

```
template<typename _Container>
auto std::begin (
    const _Container & __cont) -> decltype(__cont.begin()) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the const container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**boolalpha()**

```
ios_base & std::boolalpha (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::boolalpha)`.

References [std::ios\\_base::boolalpha](#).

**cbegin()**

```
template<typename _Container>
auto std::cbegin (
    const _Container & __cont) -> decltype(std::begin(__cont))    [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the const container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

References [begin\(\)](#).

Referenced by [std::vector<\\_Tp, \\_Alloc>::insert\(\)](#).

**cend()**

```
template<typename _Container>
auto std::cend (
    const _Container & __cont) -> decltype(std::end(__cont))    [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

References [end\(\)](#).

**const\_pointer\_cast()**

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::const_pointer_cast (
    const __shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]
const_pointer_cast
```

**crbegin()**

```
template<typename _Container>
auto std::crbegin (
    const _Container & __cont) -> decltype(std::rbegin(__cont))    [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing to the last element of the const container.



**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

References [rbegin\(\)](#).

**crend()**

```
template<typename _Container>
auto std::crend (
    const _Container & __cont) -> decltype(std::rend(__cont))    [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing one past the first element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

References [rend\(\)](#).

**data() [1/4]**

```
template<typename _Container>
auto std::data (
    _Container & __cont) -> decltype(__cont.data())    [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of a container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Referenced by [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc >::find\(\)](#).

**data() [2/4]**

```
template<typename _Tp, size_t _Nm>
_Tp * std::data (
    _Tp(&) __array[_Nm])    [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of an array.

**Parameters**

|                      |        |
|----------------------|--------|
| <code>__array</code> | Array. |
|----------------------|--------|

**data() [3/4]**

```
template<typename _Container>
auto std::data (
    const _Container & __cont) -> decltype(__cont.data())    [nodiscard], [constexpr],
[noexcept]
```

Return the data pointer of a const container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**data()** [4/4]

```
template<typename _Tp>
const _Tp * std::data (
    initializer_list< _Tp > __il) [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of an initializer list.

## Parameters

|                   |                   |
|-------------------|-------------------|
| <code>__il</code> | Initializer list. |
|-------------------|-------------------|

**dec()**

```
ios_base & std::dec (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::dec, ios_base::basefield)`.

References [std::ios\\_base::basefield](#), and [std::ios\\_base::dec](#).

**defaultfloat()**

```
ios_base & std::defaultfloat (
    ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::floatfield)`

References [std::ios\\_base::floatfield](#).

**distance()**

```
template<typename _InputIterator>
iterator_traits< _InputIterator >::difference_type std::distance (
    _InputIterator __first,
    _InputIterator __last) [inline], [nodiscard], [constexpr]
```

A generalization of pointer arithmetic.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

## Returns

The distance between them.

Returns `n` such that `__first + n == __last`. This requires that `__last` must be reachable from `__first`. Note that `n` may be negative.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

References [\\_\\_iterator\\_category\(\)](#), and [move\(\)](#).

Referenced by [\\_\\_merge\\_without\\_buffer\(\)](#), [\\_\\_sample\(\)](#), [std::deque<\\_Tp, \\_Alloc>::\\_M\\_range\\_initialize\(\)](#), [is\\_heap\(\)](#), [is\\_heap\\_until\(\)](#), [is\\_heap\\_until\(\)](#), [std::sub\\_match<const char\\*>::length\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#), [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer<node, leaf, head, inode, const\\_iterator, iterator, \\_A>::partition\\_point\(\)](#), [std::match\\_results<\\_BidirectionalIterator, polymorphic\\_allocator<sub\\_match<\\_BidirectionalIterator>>>><const char\\*>::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>><\\_BigBlock>::reserve\(\)](#), [std::list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::size\(\)](#), and [std::list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::splice\(\)](#).

### dynamic\_pointer\_cast()

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast (
    const __shared_ptr<_Tp1, _Lp> & __r) [inline], [noexcept]
dynamic_pointer_cast
```

### empty() [1/3]

```
template<typename _Container>
auto std::empty (
    const _Container & __cont) -> decltype(__cont.empty()) [nodiscard], [constexpr],
[noexcept]
```

Return whether a container is empty.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Referenced by [std::list<\\_Tp, \\_Alloc>::sort\(\)](#), and [std::list<\\_Tp, \\_Alloc>::sort\(\)](#).

### empty() [2/3]

```
template<typename _Tp, size_t _Nm>
bool std::empty (
    const _Tp(&)[_Nm]) [nodiscard], [constexpr], [noexcept]
```

Return whether an array is empty (always false).

### empty() [3/3]

```
template<typename _Tp>
bool std::empty (
    initializer_list<_Tp> __il) [nodiscard], [constexpr], [noexcept]
```

Return whether an initializer\_list is empty.

#### Parameters

|                   |                   |
|-------------------|-------------------|
| <code>__il</code> | Initializer list. |
|-------------------|-------------------|

### end() [1/4]

```
template<typename _Container>
auto std::end (
    _Container & __cont) -> decltype(__cont.end()) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**end()** [2/4]

```
template<typename _Tp, size_t _Nm>
_Tp * std::end (
    _Tp(&) __arr[_Nm]) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the array.

## Parameters

|                    |        |
|--------------------|--------|
| <code>__arr</code> | Array. |
|--------------------|--------|

**end()** [3/4]

```
template<typename _Container>
auto std::end (
    const _Container & __cont) -> decltype(__cont.end()) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**endl()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::endl (
    basic_ostream< _CharT, _Traits > & __os) [inline]
```

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple newline is desired, leading to poor buffering performance. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.↵streambuf.buffering> for more on this subject.

**ends()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::ends (
    basic_ostream< _CharT, _Traits > & __os) [inline]
```

Write a null character into the output sequence.

*Null character* is `CharT()` by definition. For `CharT` of `char`, this correctly writes the ASCII NUL character string terminator.

**fabs()**

```
template<typename _Tp>
_Tp std::fabs (
    const std::complex< _Tp > & __z) [inline]
fabs(__z) TR1 8.1.8 [tr.c99.cmplx.fabs]
```

**fixed()**

```
ios_base & std::fixed (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

References [fixed\(\)](#), [std::ios\\_base::fixed](#), and [std::ios\\_base::floatfield](#).

Referenced by [fixed\(\)](#).

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::flush (
    basic_ostream< _CharT, _Traits > & __os) [inline]
```

Flushes the output stream.

This manipulator simply calls the stream's `flush()` member function.

**frexp()**

```
template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > frexp (
    const simd< _Tp, _Abi > & __x,
    _Samesize< int, simd< _Tp, _Abi > > * __exp)
```

splits `__v` into exponent and mantissa, the sign is kept with the mantissa

The return value will be in the range `[0.5, 1.0[` The `__e` value will be an integer defining the power-of-two exponent

References [abs\(\)](#).

**from\_chars()**

```
template<typename _Tp, enable_if_t< __or< __is_standard_integer< _Tp >, is_same< char, remove_cv_t< _Tp > > >::value, int > = 0>
from_chars_result std::from_chars (
    const char * __first,
    const char * __last,
    _Tp & __value,
    int __base = 10) [constexpr]
```

`std::from_chars` for integral types.

**get()** [1/12]

```
template<typename _Tp, typename _Up>
const _Tp && std::get (
    const pair< _Tp, _Up > && __p) [constexpr], [noexcept]
```

`std::get` overloads for accessing members of `std::pair`

References [forward\(\)](#).

**get()** [2/12]

```
template<typename _Tp, typename _Up>
const _Tp & std::get (
    const pair< _Tp, _Up > & __p) [constexpr], [noexcept]
```

`std::get` overloads for accessing members of `std::pair`

**get()** [3/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
```

```
const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (
    const pair< _Tp1, _Tp2 > && __in) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair  
References [move\(\)](#).

**get()** [4/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (
    const pair< _Tp1, _Tp2 > & __in) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

**get()** [5/12]

```
template<typename _Tp, typename _Up>
const _Tp && std::get (
    const pair< _Up, _Tp > && __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair  
References [forward\(\)](#).

**get()** [6/12]

```
template<typename _Tp, typename _Up>
const _Tp & std::get (
    const pair< _Up, _Tp > & __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

**get()** [7/12]

```
template<typename _Tp, typename _Up>
_Tp && std::get (
    pair< _Tp, _Up > && __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair  
References [forward\(\)](#).

**get()** [8/12]

```
template<typename _Tp, typename _Up>
_Tp & std::get (
    pair< _Tp, _Up > & __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

**get()** [9/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (
    pair< _Tp1, _Tp2 > && __in) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair  
References [move\(\)](#).

**get()** [10/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (
    pair< _Tp1, _Tp2 > & __in) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

**get()** [11/12]

```
template<typename _Tp, typename _Up>
_Tp && std::get (
    pair< _Up, _Tp > && __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

References [forward\(\)](#).

**get()** [12/12]

```
template<typename _Tp, typename _Up>
_Tp & std::get (
    pair< _Up, _Tp > & __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

**get\_money()**

```
template<typename _MoneyT>
_Get_money< _MoneyT > std::get_money (
    _MoneyT & __mon,
    bool __intl = false) [inline]
```

Extended manipulator for extracting money.

**Parameters**

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__mon</code>  | Either long double or a specialization of <code>basic_string</code> . |
| <code>__intl</code> | A bool indicating whether international format is to be used.         |

Sent to a stream object, this manipulator extracts `__mon`.

**get\_new\_handler()**

```
new_handler std::get_new_handler () [noexcept]
```

Return the current new handler.

**get\_temporary\_buffer()**

```
template<typename _Tp>
pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (
    ptrdiff_t __len) [noexcept]
```

Allocates a temporary buffer.

**Parameters**

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__len</code> | The number of objects of type <code>Tp</code> . |
|--------------------|-------------------------------------------------|

**Returns**

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `__len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A `pair<>` is returned containing *the buffer's address and capacity (in the units of `sizeof(_Tp)`)*, or a pair of 0 values if no storage can be obtained. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the nothrow exception guarantee.

**get\_time()**

```
template<typename _CharT>
_Get_time< _CharT > std::get_time (
    std::tm * __tmb,
    const _CharT * __fmt) [inline]
```

Extended manipulator for extracting time.

This manipulator uses `time_get::get` to extract time. [ext.manip]

**Parameters**

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__tmb</code> | struct to extract the time data to. |
| <code>__fmt</code> | format string.                      |

**getline()** [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
    basic_istream< _CharT, _Traits > && __is,
    basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Read a line from an rvalue stream into a string.

References [getline\(\)](#).

**getline()** [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
    basic_istream< _CharT, _Traits > && __is,
    basic_string< _CharT, _Traits, _Alloc > & __str,
    _CharT __delim) [inline]
```

Read a line from an rvalue stream into a string.

References [getline\(\)](#).

**getline()** [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::getline (
    basic_istream< _CharT, _Traits > & __is,
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Read a line from stream into a string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |



**Returns**

Reference to the input stream.

Stores characters from `is` into `__str` until ' ' is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If end of line was encountered, it is extracted but not stored into `__str`.

References [getline\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::widen\(\)](#).

**getline()** [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::getline (
    basic_istream< _CharT, _Traits > & __is,
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    _CharT __delim)
Read a line from stream into a string.
```

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__is</code>    | Input stream.                  |
| <code>__str</code>   | Buffer to store into.          |
| <code>__delim</code> | Character marking end of line. |

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If `delim` was encountered, it is extracted but not stored into `__str`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::append\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::max\\_size\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**getline()** [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
    basic_istream< _CharT, _Traits > & __is,
    basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
Read a line from stream into a string.
```

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

**Returns**

Reference to the input stream.

Stores characters from `is` into `__str` until ' ' is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If end of line is encountered, it is extracted but not stored into `__str`.

References [getline\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::widen\(\)](#).

**getline()** [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
    basic_istream< _CharT, _Traits > & __is,
    basic_string< _CharT, _Traits, _Alloc > & __str,
    _CharT __delim)
```

Read a line from stream into a string.

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__is</code>    | Input stream.                  |
| <code>__str</code>   | Buffer to store into.          |
| <code>__delim</code> | Character marking end of line. |

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If `__delim` is encountered, it is extracted but not stored into `__str`. References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::erase\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::max\\_size\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#). Referenced by [getline\(\)](#), [getline\(\)](#), [getline\(\)](#), and [getline\(\)](#).

**hex()**

```
ios_base & std::hex (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::hex, ios_base::basefield)`.  
References [std::ios\\_base::basefield](#), [hex\(\)](#), and [std::ios\\_base::hex](#).  
Referenced by [hex\(\)](#), and [std::regex\\_traits< \\_Ch\\_type >::value\(\)](#).

**hexfloat()**

```
ios_base & std::hexfloat (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::fixed|ios_base::scientific, ios_base::floatfield)`.  
References [std::ios\\_base::fixed](#), [std::ios\\_base::floatfield](#), and [std::ios\\_base::scientific](#).

**internal()**

```
ios_base & std::internal (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.  
References [std::ios\\_base::adjustfield](#), and [std::ios\\_base::internal](#).

**isalnum()**

```
template<typename _CharT>
bool std::isalnum (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::alnum, __c)`.  
References [use\\_facet\(\)](#).

**isalpha()**

```
template<typename _CharT>
bool std::isalpha (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.

References [use\\_facet\(\)](#).

**isblank()**

```
template<typename _CharT>
bool std::isblank (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::blank, __c)`.

References [use\\_facet\(\)](#).

**isctrl()**

```
template<typename _CharT>
bool std::isctrl (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

References [use\\_facet\(\)](#).

**isdigit()**

```
template<typename _CharT>
bool std::isdigit (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

References [use\\_facet\(\)](#).

**isgraph()**

```
template<typename _CharT>
bool std::isgraph (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

References [use\\_facet\(\)](#).

**islower()**

```
template<typename _CharT>
bool std::islower (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

References [use\\_facet\(\)](#).

**isprint()**

```
template<typename _CharT>
bool std::isprint (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::print, __c)`.

References [use\\_facet\(\)](#).

**ispunct()**

```
template<typename _CharT>
bool std::ispunct (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

References [use\\_facet\(\)](#).

**isspace()**

```
template<typename _CharT>
bool std::isspace (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::space, __c)`.

References [use\\_facet\(\)](#).

**isupper()**

```
template<typename _CharT>
bool std::isupper (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

References [use\\_facet\(\)](#).

**isxdigit()**

```
template<typename _CharT>
bool std::isxdigit (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

References [use\\_facet\(\)](#).

**left()**

```
ios_base & std::left (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

References [std::ios\\_base::adjustfield](#), and [std::ios\\_base::left](#).

**noboolalpha()**

```
ios_base & std::noboolalpha (
    ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::boolalpha)`.  
References [std::ios\\_base::boolalpha](#).

### **noshowbase()**

```
ios_base & std::noshowbase (  
    ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::showbase)`.  
References [std::ios\\_base::showbase](#).

### **noshowpoint()**

```
ios_base & std::noshowpoint (  
    ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::showpoint)`.  
References [std::ios\\_base::showpoint](#).

### **noshowpos()**

```
ios_base & std::noshowpos (  
    ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::showpos)`.  
References [std::ios\\_base::showpos](#).

### **noskipws()**

```
ios_base & std::noskipws (  
    ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::skipws)`.  
References [std::ios\\_base::skipws](#).

### **nounitbuf()**

```
ios_base & std::nounitbuf (  
    ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::unitbuf)`.  
References [std::ios\\_base::unitbuf](#).

### **nouppercase()**

```
ios_base & std::nouppercase (  
    ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::uppercase)`.  
References [std::ios\\_base::uppercase](#).

### **oct()**

```
ios_base & std::oct (  
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::oct, ios_base::basefield)`.  
References [std::ios\\_base::basefield](#), and [std::ios\\_base::oct](#).  
Referenced by [std::regex\\_traits<\\_Ch\\_type>::value\(\)](#).

**operator"!="() [1/5]**

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator!= (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator==.

**operator"!="() [2/5]**

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator!= (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(x == y).

**operator"!="() [3/5]**

```
template<typename _Tp, typename _Seq>
bool std::operator!= (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator==.

**operator"!="() [4/5]**

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator!= (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(x == y).

**operator"!="() [5/5]**

```
template<typename _Tp, typename _Seq>
bool std::operator!= (
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator==.

**operator&()**

```
template<size_t _Nb>
bitset< _Nb > std::operator& (
    const bitset< _Nb > & __x,
    const bitset< _Nb > & __y) [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

**Parameters**

|                          |                                   |
|--------------------------|-----------------------------------|
| $\leftrightarrow$<br>__x | A bitset.                         |
| $\leftrightarrow$<br>__y | A bitset of the same size as __x. |

**Returns**

A new bitset.

These should be self-explanatory.

**operator+() [1/5]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
    _CharT __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate character and string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with `__lhs` followed by `__rhs`.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**operator+() [2/5]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
    const _CharT * __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate C string and string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with value of `__lhs` followed by `__rhs`.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**operator+() [3/5]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    _CharT __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate string and character.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with `__lhs` followed by `__rhs`.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::get\\_allocator\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).

**operator+()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
    const basic_string<_CharT, _Traits, _Alloc> & __lhs,
    const _CharT * __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate string and C string.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with `__lhs` followed by `__rhs`.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::get\\_allocator\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).

**operator+()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
    const basic_string<_CharT, _Traits, _Alloc> & __lhs,
    const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate two strings.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with value of `__lhs` followed by `__rhs`.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::get\\_allocator\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).



**operator<()** [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator< (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Multimap ordering relation.

**Parameters**

|                          |                                     |
|--------------------------|-------------------------------------|
| $\leftrightarrow$<br>__x | A multimap.                         |
| $\leftrightarrow$<br>__y | A multimap of the same type as __x. |

**Returns**

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with <. See std::lexicographical\_compare() for how the determination is made.

**operator<()** [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator< (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Multiset ordering relation.

**Parameters**

|                          |                                     |
|--------------------------|-------------------------------------|
| $\leftrightarrow$<br>__x | A multiset.                         |
| $\leftrightarrow$<br>__y | A multiset of the same type as __x. |

**Returns**

True iff \_\_x is lexicographically less than \_\_y.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with <. See std::lexicographical\_compare() for how the determination is made.

**operator<()** [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator< (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Queue ordering relation.

## Parameters

|       |                                   |
|-------|-----------------------------------|
| $\_x$ | A queue.                          |
| $\_y$ | A queue of the same type as $x$ . |

## Returns

True iff  $\_x$  is lexicographically less than  $\_y$ .

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with  $<$ , and `std::lexicographical_compare()` is usually used to make the determination.

References [std::queue<\\_Tp, \\_Sequence>::c](#).

**operator<()** [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator< (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Set ordering relation.

## Parameters

|       |                                 |
|-------|---------------------------------|
| $\_x$ | A set.                          |
| $\_y$ | A set of the same type as $x$ . |

## Returns

True iff  $\_x$  is lexicographically less than  $\_y$ .

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with  $<$ . See `std::lexicographical_compare()` for how the determination is made.

**operator<()** [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator< (
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Stack ordering relation.

## Parameters

|       |                                   |
|-------|-----------------------------------|
| $\_x$ | A stack.                          |
| $\_y$ | A stack of the same type as $x$ . |

**Returns**

True iff  $x$  is lexicographically less than  $__y$ .

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with  $<$ , and `std::lexicographical_compare()` is usually used to make the determination.

**operator<<()** [1/24]

```
template<typename _Ostream, typename _Tp>
__rvalue_stream_insertion_t< _Ostream, _Tp > std::operator<< (
    _Ostream && __os,
    const _Tp & __x) [inline]
```

Generic inserter for rvalue stream.

**Parameters**

|                   |                                           |
|-------------------|-------------------------------------------|
| <code>__os</code> | An input stream.                          |
| <code>__x</code>  | A reference to the object being inserted. |

**Returns**

`__os`

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

**operator<<()** [2/24]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_ostream< _CharT, _Traits > & std::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Write string to a stream.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__os</code>  | Output stream.       |
| <code>__str</code> | String to write out. |

**Returns**

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

**operator<<()** [3/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_ostream< _CharT, _Traits > & std::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Write string to a stream.

## Parameters

|                    |                      |
|--------------------|----------------------|
| <code>__os</code>  | Output stream.       |
| <code>__str</code> | String to write out. |

## Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

**operator<<()** [4/24]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
    basic_ostream< _CharT, _Traits > & __out,
    _CharT __c) [inline]
```

Character inserters.

## Parameters

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

## Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [5/24]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
    basic_ostream< _CharT, _Traits > & __out,
    char __c) [inline]
```

Character inserters.

## Parameters

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

## Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [6/24]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
    basic_ostream< _CharT, _Traits > & __out,
    const _CharT * __s) [inline]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

out

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [7/24]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
    basic_ostream< _CharT, _Traits > & __out,
    const char * __s)
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

out

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [8/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & ,
    char16_t ) [delete]
```

Character inserters.

## Parameters

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

## Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [9/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & ,
    char32_t ) [delete]
```

Character inserters.

## Parameters

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

## Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [10/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & ,
    const char16_t * ) [delete]
```

String inserters.

## Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

## Returns

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [11/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & ,
    const char32_t * ) [delete]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [12/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & ,
    const wchar_t * ) [delete]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [13/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & ,
    wchar_t ) [delete]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [14/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & __out,
    char __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [15/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & __out,
    const char * __s) [inline]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |



**Returns**

out

**Precondition**

\_\_s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [16/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & __out,
    const signed char * __s) [inline]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

out

**Precondition**

\_\_s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [17/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & __out,
    const unsigned char * __s) [inline]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

out

**Precondition**

\_\_s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [18/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & __out,
    signed char __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [19/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
    basic_ostream< char, _Traits > & __out,
    unsigned char __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [20/24]

```
template<typename _Traits>
basic_ostream< wchar_t, _Traits > & std::operator<< (
    basic_ostream< wchar_t, _Traits > & ,
    char16_t ) [delete]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [21/24]

```
template<typename _Traits>
basic_ostream< wchar_t, _Traits > & std::operator<< (
    basic_ostream< wchar_t, _Traits > & ,
    char32_t ) [delete]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [22/24]

```
template<typename _Traits>
basic_ostream< wchar_t, _Traits > & std::operator<< (
    basic_ostream< wchar_t, _Traits > & ,
    const char16_t * ) [delete]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

out

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [23/24]

```
template<typename _Traits>
basic_ostream< wchar_t, _Traits > & std::operator<< (
    basic_ostream< wchar_t, _Traits > & ,
    const char32_t * ) [delete]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [24/24]

```
template<class _CharT, class _Traits, size_t _Nb>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const bitset< _Nb > & __x)
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept 0 and 1 characters, and will only extract as many digits as the bitset will hold.

Referenced by `std::shared_ptr< _State_base >::operator<<()`.

**operator<=()** [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator<= (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on `operator<`.

**operator<=()** [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator<= (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns `!(y < x)`

**operator<=()** [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator<= (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator<=()** [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator<= (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(y < x)

**operator<=()** [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator<= (
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator<=>()** [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
auto std::operator<=> (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const _CharT * __rhs) -> decltype(__detail::__char_traits_cmp_cat<_Traits>(0)) [nodiscard],
[constexpr], [noexcept]
```

Three-way comparison of a string and a C string.

**Parameters**

|                    |                           |
|--------------------|---------------------------|
| <code>__lhs</code> | A string.                 |
| <code>__rhs</code> | A null-terminated string. |

**Returns**

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator<=>()** [2/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
auto std::operator<=> (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs) -> decltype(__detail::__char_traits_cmp_cat<_Traits>(0)) [nodiscard], [constexpr], [noexcept]
```

Three-way comparison of a string and a C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | A string. |
|--------------------|-----------|

|                    |                           |
|--------------------|---------------------------|
| <code>__rhs</code> | A null-terminated string. |
|--------------------|---------------------------|

**Returns**

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator<=>() [3/7]**

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
    const deque< _Tp, _Alloc > & __x,
    const deque< _Tp, _Alloc > & __y) [inline], [nodiscard]
```

Deque ordering relation.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__x</code> | A deque.                                       |
| <code>__y</code> | A deque of the same type as <code>__x</code> . |

**Returns**

A value indicating whether `__x` is less than, equal to, greater than, or incomparable with `__y`.

See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like `<` and `>=` etc.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, and `std::deque< _Tp, _Alloc >::size()`.

**operator<=>() [4/7]**

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
    const forward_list< _Tp, _Alloc > & __x,
    const forward_list< _Tp, _Alloc > & __y) [inline], [nodiscard]
```

Forward list ordering relation.

**Parameters**

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <code>__x</code> | A <code>forward_list</code> .                                      |
| <code>__y</code> | A <code>forward_list</code> of the same type as <code>__x</code> . |

**Returns**

A value indicating whether `__x` is less than, equal to, greater than, or incomparable with `__y`.

See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like `<` and `>=` etc.

**operator<=>()** [5/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
    const list< _Tp, _Alloc > & __x,
    const list< _Tp, _Alloc > & __y) [inline], [nodiscard]
```

List ordering relation.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| $\leftarrow$<br>__x | A list.                         |
| $\leftarrow$<br>__y | A list of the same type as __x. |

**Returns**

A value indicating whether \_\_x is less than, equal to, greater than, or incomparable with \_\_y.

See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::end()`, and `std::list< _Tp, _Alloc >::size()`.

**operator<=>()** [6/7]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
__detail::__synth3way_t< pair< const _Key, _Tp > > std::operator<=> (
    const map< _Key, _Tp, _Compare, _Alloc > & __x,
    const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Map ordering relation.

**Parameters**

|                     |                              |
|---------------------|------------------------------|
| $\leftarrow$<br>__x | A map.                       |
| $\leftarrow$<br>__y | A map of the same type as x. |

**Returns**

A value indicating whether \_\_x is less than, equal to, greater than, or incomparable with \_\_y.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.

**operator<=>()** [7/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
    const vector< _Tp, _Alloc > & __x,
    const vector< _Tp, _Alloc > & __y) [nodiscard], [constexpr]
```

Vector ordering relation.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_x$ | A vector.                            |
| $\_y$ | A vector of the same type as $\_x$ . |

## Returns

A value indicating whether  $\_x$  is less than, equal to, greater than, or incomparable with  $\_y$ .

See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like `<` and `>=` etc.

References [std::vector<\\_Tp, \\_Alloc>::begin\(\)](#), [std::vector<\\_Tp, \\_Alloc>::end\(\)](#), and [lexicographical\\_compare\\_three\\_way\(\)](#).

**operator==(** [1/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::operator==(
    const basic\_string< _CharT, _Traits, _Alloc > & __lhs,
    const _CharT * __rhs) [inline], [nodiscard], [constexpr]
```

Test equivalence of string and C string.

## Parameters

|         |           |
|---------|-----------|
| $\_lhs$ | String.   |
| $\_rhs$ | C string. |

## Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::data\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).

**operator==(** [2/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::operator==(
    const basic\_string< _CharT, _Traits, _Alloc > & __lhs,
    const basic\_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [nodiscard], [constexpr],
[noexcept]
```

Test equivalence of two strings.

## Parameters

|         |                |
|---------|----------------|
| $\_lhs$ | First string.  |
| $\_rhs$ | Second string. |

## Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.



**operator==()** [3/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
    const deque< _Tp, _Alloc > & __x,
    const deque< _Tp, _Alloc > & __y) [inline], [nodiscard]
```

Deque equality comparison.

**Parameters**

|                          |                                  |
|--------------------------|----------------------------------|
| $\leftrightarrow$<br>__x | A deque.                         |
| $\leftrightarrow$<br>__y | A deque of the same type as __x. |

**Returns**

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

**operator==()** [4/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly)
```

Forward list equality comparison.

**Parameters**

|                           |                                          |
|---------------------------|------------------------------------------|
| $\leftrightarrow$<br>__lx | A forward_list                           |
| $\leftrightarrow$<br>__ly | A forward_list of the same type as __lx. |

**Returns**

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

References [std::forward\\_list< \\_Tp, \\_Alloc >::cbegin\(\)](#), and [std::forward\\_list< \\_Tp, \\_Alloc >::cend\(\)](#).

**operator==()** [5/14]

```
template<typename _StateT>
bool std::operator== (
    const fpos< _StateT > & __lhs,
    const fpos< _StateT > & __rhs) [inline]
```

Test if equivalent to another position.

**operator==()** [6/14]

```
template<typename _Res, typename... _Args>
bool std::operator== (
    const function< _Res(_Args...)> & __f,
    nullptr_t ) [inline], [noexcept]
```

Test whether a polymorphic function object wrapper is empty.

**Returns**

true if the wrapper has no target, false otherwise

This function will not throw exceptions.

**operator==()** [7/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
    const list< _Tp, _Alloc > & __x,
    const list< _Tp, _Alloc > & __y) [inline], [nodiscard]
```

List equality comparison.

**Parameters**

|                          |                                 |
|--------------------------|---------------------------------|
| $\leftrightarrow$<br>__x | A list.                         |
| $\leftrightarrow$<br>__y | A list of the same type as __x. |

**Returns**

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

**operator==()** [8/14]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator== (
    const map< _Key, _Tp, _Compare, _Alloc > & __x,
    const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Map equality comparison.

**Parameters**

|                          |                              |
|--------------------------|------------------------------|
| $\leftrightarrow$<br>__x | A map.                       |
| $\leftrightarrow$<br>__y | A map of the same type as x. |

**Returns**

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

**operator==()** [9/14]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator== (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Multimap equality comparison.

**Parameters**

|                          |                                     |
|--------------------------|-------------------------------------|
| $\leftrightarrow$<br>__x | A multimap.                         |
| $\leftrightarrow$<br>__y | A multimap of the same type as __x. |

**Returns**

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

**operator==()** [10/14]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator== (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Multiset equality comparison.

**Parameters**

|                          |                                     |
|--------------------------|-------------------------------------|
| $\leftrightarrow$<br>__x | A multiset.                         |
| $\leftrightarrow$<br>__y | A multiset of the same type as __x. |

**Returns**

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

**operator==()** [11/14]

```
template<typename _Tp, typename _Seq>
bool std::operator== (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Queue equality comparison.

## Parameters

|       |                                     |
|-------|-------------------------------------|
| $\_x$ | A queue.                            |
| $\_y$ | A queue of the same type as $\_x$ . |

## Returns

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

References [std::queue<\\_Tp, \\_Sequence>::c](#).

**operator==()** [12/14]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator== (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Set equality comparison.

## Parameters

|       |                                   |
|-------|-----------------------------------|
| $\_x$ | A set.                            |
| $\_y$ | A set of the same type as $\_x$ . |

## Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

**operator==()** [13/14]

```
template<typename _Tp, typename _Seq>
bool std::operator== (
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Stack equality comparison.

## Parameters

|       |                                     |
|-------|-------------------------------------|
| $\_x$ | A stack.                            |
| $\_y$ | A stack of the same type as $\_x$ . |

**Returns**

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

**operator==()** [14/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
    const vector< _Tp, _Alloc > & __x,
    const vector< _Tp, _Alloc > & __y) [inline], [nodiscard], [constexpr]
```

Vector equality comparison.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| $\leftarrow$<br>_x | A vector.                         |
| $\leftarrow$<br>_y | A vector of the same type as __x. |

**Returns**

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

References `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, and `std::vector< _Tp, _Alloc >::size()`.

**operator>()** [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator> (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on `operator<`.

**operator>()** [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator> (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns  $y < x$ .

**operator>()** [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator> (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on `operator<`.

**operator>()** [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator> (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns  $y < x$ .

**operator>()** [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator> (
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator>=()** [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator>= (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator<.

**operator>=()** [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator>= (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns  $!(x < y)$

**operator>=()** [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator>= (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator>=()** [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator>= (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns  $!(x < y)$

**operator>=()** [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator>= (
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator>>()** [1/10]

```
template<typename _Istream, typename _Tp>
__rvalue_stream_extraction_t< _Istream, _Tp > std::operator>> (
    _Istream && __is,
    _Tp && __x) [inline]
```

Generic extractor for rvalue stream.

**Parameters**

|                   |                                       |
|-------------------|---------------------------------------|
| <code>__is</code> | An input stream.                      |
| <code>__x</code>  | A reference to the extraction target. |

**Returns**

`__is`

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

**operator>>()** [2/10]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::operator>> (
    basic_istream< _CharT, _Traits > & __in,
    _CharT & __c)
```

Character extractors.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__in</code> | An input stream.       |
| <code>__c</code>  | A character reference. |

**Returns**

`in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdstate()` and `std::basic_ios< _CharT, _Traits >::setstate()`.

**operator>>()** [3/10]

```
template<typename _CharT, typename _Traits, size_t _Num>
basic_istream< _CharT, _Traits > & std::operator>> (
    basic_istream< _CharT, _Traits > & __in,
    _CharT(&) __s[_Num]) [inline]
```

Character string extractors.

## Parameters

|                         |                                                            |
|-------------------------|------------------------------------------------------------|
| <code>_↔<br/>_in</code> | An input stream.                                           |
| <code>_↔<br/>_s</code>  | A character array (or a pointer to an array before C++20). |

## Returns

`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

**operator>>()** [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::operator>> (
    basic_istream< _CharT, _Traits > & __is,
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)
```

Read stream into a string.

## Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

## Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

References `std::ios_base::getloc()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `use_facet()`, and `std::ios_base::width()`.



**operator>>()** [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::operator>> (
    basic_istream< _CharT, _Traits > & __is,
    basic_string< _CharT, _Traits, _Alloc > & __str)
```

Read stream into a string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::erase()`, `std::ios_base::getloc()`, `std::basic_string< _CharT, _Traits, _Alloc >::max_size()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `use_facet()`, and `std::ios_base::width()`.

**operator>>()** [6/10]

```
template<class _Traits>
basic_istream< char, _Traits > & std::operator>> (
    basic_istream< char, _Traits > & __in,
    signed char & __c) [inline]
```

Character extractors.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__in</code> | An input stream.       |
| <code>__c</code>  | A character reference. |

**Returns**

`in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

**operator>>()** [7/10]

```
template<class _Traits, size_t _Num>
basic_istream< char, _Traits > & std::operator>> (
    basic_istream< char, _Traits > & __in,
    signed char(&) __s[_Num]) [inline]
```

Character string extractors.

## Parameters

|                         |                                                            |
|-------------------------|------------------------------------------------------------|
| <code>_↔<br/>_in</code> | An input stream.                                           |
| <code>_↔<br/>_s</code>  | A character array (or a pointer to an array before C++20). |

## Returns

`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

**operator>>()** [8/10]

```
template<class _Traits>
basic_istream< char, _Traits > & std::operator>> (
    basic_istream< char, _Traits > & __in,
    unsigned char & __c) [inline]
```

Character extractors.

## Parameters

|                         |                        |
|-------------------------|------------------------|
| <code>_↔<br/>_in</code> | An input stream.       |
| <code>_↔<br/>_c</code>  | A character reference. |

## Returns

`in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

**operator>>()** [9/10]

```
template<class _Traits, size_t _Num>
basic_istream< char, _Traits > & std::operator>> (
    basic_istream< char, _Traits > & __in,
    unsigned char(&) __s[_Num]) [inline]
```

Character string extractors.

**Parameters**

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__in</code> | An input stream.                                           |
| <code>__s</code>  | A character array (or a pointer to an array before C++20). |

**Returns**

`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

**operator>>()** [10/10]

```
template<class _CharT, class _Traits, size_t _Nb>
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    bitset< _Nb > & __x)
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept `0` and `1` characters, and will only extract as many digits as the bitset will hold.

**operator^()**

```
template<size_t _Nb>
bitset< _Nb > std::operator^ (
    const bitset< _Nb > & __x,
    const bitset< _Nb > & __y) [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_x$ | A bitset.                            |
| $\_y$ | A bitset of the same size as $\_x$ . |

**Returns**

A new bitset.

These should be self-explanatory.

**operator" | ()**

```
template<size_t _Nb>
bitset< _Nb > std::operator| (
    const bitset< _Nb > & __x,
    const bitset< _Nb > & __y) [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_x$ | A bitset.                            |
| $\_y$ | A bitset of the same size as $\_x$ . |

**Returns**

A new bitset.

These should be self-explanatory.

**put\_money()**

```
template<typename _MoneyT>
_Put_money< _MoneyT > std::put_money (
    const _MoneyT & __mon,
    bool __intl = false) [inline]
```

Extended manipulator for inserting money.

**Parameters**

|          |                                                                       |
|----------|-----------------------------------------------------------------------|
| $\_mon$  | Either long double or a specialization of <code>basic_string</code> . |
| $\_intl$ | A bool indicating whether international format is to be used.         |

Sent to a stream object, this manipulator inserts  $\_mon$ .

**put\_time()**

```
template<typename _CharT>
_Put_time< _CharT > std::put_time (
```

```
const std::tm * __tmb,  
const _CharT * __fmt) [inline]
```

Extended manipulator for formatting time.

This manipulator uses `time_put::put` to format time. [ext.manip]

## Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__tmb</code> | struct tm time data to format. |
| <code>__fmt</code> | format string.                 |

**rbegin()** [1/4]

```
template<typename _Container>
auto std::rbegin (
    _Container & __cont) -> decltype(__cont.rbegin())    [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Referenced by [crbegin\(\)](#).

**rbegin()** [2/4]

```
template<typename _Tp, size_t _Nm>
reverse_iterator< _Tp * > std::rbegin (
    _Tp(&) __arr[_Nm])    [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the array.

## Parameters

|                    |        |
|--------------------|--------|
| <code>__arr</code> | Array. |
|--------------------|--------|

**rbegin()** [3/4]

```
template<typename _Container>
auto std::rbegin (
    const _Container & __cont) -> decltype(__cont.rbegin())    [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing to the last element of the const container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**rbegin()** [4/4]

```
template<typename _Tp>
reverse_iterator< const _Tp * > std::rbegin (
    initializer_list< _Tp > __il)    [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the initializer\_list.

**Parameters**

|                                                                             |                   |
|-----------------------------------------------------------------------------|-------------------|
| <a href="#"><math>\leftrightarrow</math></a><br><a href="#"><i>__il</i></a> | initializer_list. |
|-----------------------------------------------------------------------------|-------------------|

**rend() [1/4]**

```
template<typename _Container>
auto std::rend (
    _Container & __cont) -> decltype(__cont.rend())    [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the container.

**Parameters**

|                               |            |
|-------------------------------|------------|
| <a href="#"><i>__cont</i></a> | Container. |
|-------------------------------|------------|

Referenced by [crend\(\)](#).

**rend() [2/4]**

```
template<typename _Tp, size_t _Nm>
reverse_iterator< _Tp * > std::rend (
    _Tp(&) __arr[_Nm])    [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the array.

**Parameters**

|                              |        |
|------------------------------|--------|
| <a href="#"><i>__arr</i></a> | Array. |
|------------------------------|--------|

**rend() [3/4]**

```
template<typename _Container>
auto std::rend (
    const _Container & __cont) -> decltype(__cont.rend())    [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the const container.

**Parameters**

|                               |            |
|-------------------------------|------------|
| <a href="#"><i>__cont</i></a> | Container. |
|-------------------------------|------------|

**rend() [4/4]**

```
template<typename _Tp>
reverse_iterator< const _Tp * > std::rend (
    initializer_list< _Tp > __il)    [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the initializer\_list.

**Parameters**

|                                                                             |                   |
|-----------------------------------------------------------------------------|-------------------|
| <a href="#"><math>\leftrightarrow</math></a><br><a href="#"><i>__il</i></a> | initializer_list. |
|-----------------------------------------------------------------------------|-------------------|

**replace\_copy()**

```
template<typename _InputIterator, typename _OutputIterator, typename _Tp>
_OutputIterator std::replace_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    const _Tp & __old_value,
    const _Tp & __new_value) [inline], [constexpr]
```

Copy a sequence, replacing each element of one value with another value.

**Parameters**

|                          |                           |
|--------------------------|---------------------------|
| <code>__first</code>     | An input iterator.        |
| <code>__last</code>      | An input iterator.        |
| <code>__result</code>    | An output iterator.       |
| <code>__old_value</code> | The value to be replaced. |
| <code>__new_value</code> | The replacement value.    |

**Returns**

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range `[__first,__last)` to the output range `[__result,__result+(__last-__first))` replacing elements equal to `__old_value` with `__new_value`.

**resetiosflags()**

```
_Resetiosflags std::resetiosflags (
    ios_base::fmtflags __mask) [inline]
```

Manipulator for `setf`.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__mask</code> | A format flags mask. |
|---------------------|----------------------|

Sent to a stream object, this manipulator resets the specified flags, via `stream.setf(0,__mask)`.

**return\_temporary\_buffer()**

```
template<typename _Tp>
void std::return_temporary_buffer (
    _Tp * __p) [inline]
```

The companion to `get_temporary_buffer()`.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__p</code> | A buffer previously allocated by <code>get_temporary_buffer</code> . |
|------------------|----------------------------------------------------------------------|

**Returns**

None.

Frees the memory pointed to by `__p`.



**right()**

```
ios_base & std::right (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

References [std::ios\\_base::adjustfield](#), and [std::ios\\_base::right](#).

**sample()**

```
template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _URBG↵
UniformRandomBitGenerator>
_SampleIterator std::sample (
    _PopulationIterator __first,
    _PopulationIterator __last,
    _SampleIterator __out,
    _Distance __n,
    _UniformRandomBitGenerator && __g)
```

Take a random sample from a population.

References [forward\(\)](#).

**scientific()**

```
ios_base & std::scientific (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

References [std::ios\\_base::floatfield](#), [std::ios\\_base::scientific](#), and [scientific\(\)](#).

Referenced by [scientific\(\)](#).

**search()**

```
template<typename _ForwardIterator, typename _Searcher>
_FowardIterator std::search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Searcher & __searcher) [inline], [nodiscard], [constexpr]
```

Search a sequence using a Searcher object.

**Parameters**

|                         |                     |
|-------------------------|---------------------|
| <code>__first</code>    | A forward iterator. |
| <code>__last</code>     | A forward iterator. |
| <code>__searcher</code> | A callable object.  |

**Returns**

```
__searcher(__first, __last).first
```

**set\_new\_handler()**

```
new_handler std::set_new_handler (
    new_handler ) throw ( )
```

Takes a replacement handler as the argument, returns the previous handler.

**setbase()**

```
_Setbase std::setbase (  
    int __base) [inline]
```

Manipulator for `setf`.

**Parameters**

|                     |                 |
|---------------------|-----------------|
| <code>__base</code> | A numeric base. |
|---------------------|-----------------|

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when `base` is 8, 10, or 16, accordingly, and to 0 if `__base` is any other value.

**setfill()**

```
template<typename _CharT>  
_Setfill< _CharT > std::setfill (  
    _CharT __c) [inline]
```

Manipulator for `fill`.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The new fill character. |
|------------------|-------------------------|

Sent to a stream object, this manipulator calls `fill(__c)` for that object.

**setiosflags()**

```
_Setiosflags std::setiosflags (  
    ios_base::fmtflags __mask) [inline]
```

Manipulator for `setf`.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__mask</code> | A format flags mask. |
|---------------------|----------------------|

Sent to a stream object, this manipulator sets the format flags to `__mask`.

**setprecision()**

```
_Setprecision std::setprecision (  
    int __n) [inline]
```

Manipulator for `precision`.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__n</code> | The new precision. |
|------------------|--------------------|

Sent to a stream object, this manipulator calls `precision(__n)` for that object.

**setw()**

```
_Setw std::setw (
    int __n) [inline]
```

Manipulator for width.

## Parameters

|                  |                |
|------------------|----------------|
| <code>__n</code> | The new width. |
|------------------|----------------|

Sent to a stream object, this manipulator calls `width(__n)` for that object.

**showbase()**

```
ios_base & std::showbase (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::showbase)`.

References [std::ios\\_base::showbase](#).

**showpoint()**

```
ios_base & std::showpoint (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::showpoint)`.

References [std::ios\\_base::showpoint](#).

**showpos()**

```
ios_base & std::showpos (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::showpos)`.

References [std::ios\\_base::showpos](#).

**size()** [1/2]

```
template<typename _Container>
auto std::size (
    const _Container & __cont) -> decltype(__cont.size()) [nodiscard], [constexpr],
[noexcept]
```

Return the size of a container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Referenced by [std::deque<\\_Tp, \\_Alloc>::\\_M\\_new\\_elements\\_at\\_back\(\)](#), [std::deque<\\_Tp, \\_Alloc>::\\_M\\_new\\_elements\\_at\\_front\(\)](#), [std::deque<\\_Tp, \\_Alloc>::\\_M\\_push\\_back\\_aux\(\)](#), [std::deque<\\_Tp, \\_Alloc>::\\_M\\_push\\_front\\_aux\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::find\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::find\\_first\\_not\\_of\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::find\\_first\\_of\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::find\\_last\\_not\\_of\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::find\\_last\\_of\(\)](#), [std::deque<\\_Tp, \\_Alloc>::operator=\(\)](#), [std::vector<\\_Tp, \\_Alloc>::operator=\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::reserve\(\)](#), [std::vector<\\_Tp, \\_Alloc>::reserve\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::reserve\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::rfind\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::rfind\(\)](#).

**size()** [2/2]

```
template<typename _Tp, size_t _Nm>
size_t std::size (
    const _Tp(&)[_Nm]) [nodiscard], [constexpr], [noexcept]
```

Return the size of an array.

**skipws()**

```
ios_base & std::skipws (  
    ios_base & __base) [inline]
```

Calls base.setf(ios\_base::skipws).

References [std::ios\\_base::skipws](#).

**static\_pointer\_cast()**

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>  
__shared_ptr< _Tp, _Lp > std::static_pointer_cast (  
    const __shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]
```

static\_pointer\_cast

**swap() [1/19]**

```
template<class _CharT, class _Traits>  
void std::swap (  
    basic_filebuf< _CharT, _Traits > & __x,  
    basic_filebuf< _CharT, _Traits > & __y) [inline]
```

Swap specialization for filebufs.

**swap() [2/19]**

```
template<class _CharT, class _Traits>  
void std::swap (  
    basic_fstream< _CharT, _Traits > & __x,  
    basic_fstream< _CharT, _Traits > & __y) [inline]
```

Swap specialization for fstreams.

**swap() [3/19]**

```
template<class _CharT, class _Traits>  
void std::swap (  
    basic_ifstream< _CharT, _Traits > & __x,  
    basic_ifstream< _CharT, _Traits > & __y) [inline]
```

Swap specialization for ifstreams.

**swap() [4/19]**

```
template<class _CharT, class _Traits, class _Allocator>  
void std::swap (  
    basic_istringstream< _CharT, _Traits, _Allocator > & __x,  
    basic_istringstream< _CharT, _Traits, _Allocator > & __y) [inline]
```

Swap specialization for istringstreams.

**swap() [5/19]**

```
template<class _CharT, class _Traits>  
void std::swap (  
    basic_ofstream< _CharT, _Traits > & __x,  
    basic_ofstream< _CharT, _Traits > & __y) [inline]
```

Swap specialization for ofstreams.

**swap()** [6/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
    basic_ostringstream< _CharT, _Traits, _Allocator > & __x,
    basic_ostringstream< _CharT, _Traits, _Allocator > & __y) [inline]
```

Swap specialization for ostringstreams.

**swap()** [7/19]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::swap (
    basic_string< _CharT, _Traits, _Alloc > & __lhs,
    basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [constexpr], [noexcept]
```

Swap contents of two strings.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

Exchanges the contents of `__lhs` and `__rhs` in constant time.

**swap()** [8/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
    basic_stringbuf< _CharT, _Traits, _Allocator > & __x,
    basic_stringbuf< _CharT, _Traits, _Allocator > & __y) [inline], [noexcept]
```

Swap specialization for stringbufs.

**swap()** [9/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
    basic_stringstream< _CharT, _Traits, _Allocator > & __x,
    basic_stringstream< _CharT, _Traits, _Allocator > & __y) [inline]
```

Swap specialization for stringstreams.

**swap()** [10/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
    deque< _Tp, _Alloc > & __x,
    deque< _Tp, _Alloc > & __y) [inline], [noexcept]
```

See `std::deque::swap()`.

**swap()** [11/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
    forward_list< _Tp, _Alloc > & __lx,
    forward_list< _Tp, _Alloc > & __ly) [inline], [noexcept]
```

See `std::forward_list::swap()`.

**swap()** [12/19]

```
template<typename _Res, typename... _Args>
void std::swap (
    function< _Res(_Args...)> & __x,
    function< _Res(_Args...)> & __y) [inline], [noexcept]
```

Swap the targets of two polymorphic function object wrappers.

This function will not throw exceptions.

**swap()** [13/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
    list< _Tp, _Alloc > & __x,
    list< _Tp, _Alloc > & __y) [inline], [noexcept]
```

See [std::list::swap\(\)](#).

**swap()** [14/19]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
void std::swap (
    map< _Key, _Tp, _Compare, _Alloc > & __x,
    map< _Key, _Tp, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See [std::map::swap\(\)](#).

References [std::pair<\\_T1, \\_T2>::swap\(\)](#).

**swap()** [15/19]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
void std::swap (
    multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See [std::multimap::swap\(\)](#).

References [std::multimap<\\_Key, \\_Tp, \\_Compare, \\_Alloc>::swap\(\)](#).

**swap()** [16/19]

```
template<typename _Key, typename _Compare, typename _Alloc>
void std::swap (
    multiset< _Key, _Compare, _Alloc > & __x,
    multiset< _Key, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See [std::multiset::swap\(\)](#).

References [std::multiset<\\_Key, \\_Compare, \\_Alloc>::swap\(\)](#).

**swap()** [17/19]

```
template<typename _Key, typename _Compare, typename _Alloc>
void std::swap (
    set< _Key, _Compare, _Alloc > & __x,
    set< _Key, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See [std::set::swap\(\)](#).

References [std::set<\\_Key, \\_Compare, \\_Alloc>::swap\(\)](#).

**swap()** [18/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
    vector< _Tp, _Alloc > & __x,
    vector< _Tp, _Alloc > & __y) [inline], [constexpr], [noexcept]
```

See `std::vector::swap()`.

**tolower()**

```
template<typename _CharT>
_CharT std::tolower (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype::tolower(__c)`.

References [tolower\(\)](#), and [use\\_facet\(\)](#).

Referenced by [tolower\(\)](#).

**toupper()**

```
template<typename _CharT>
_CharT std::toupper (
    _CharT __c,
    const locale & __loc) [inline]
```

Convenience interface to `ctype::toupper(__c)`.

References [toupper\(\)](#), and [use\\_facet\(\)](#).

Referenced by [toupper\(\)](#).

**unitbuf()**

```
ios_base & std::unitbuf (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::unitbuf)`.

References [std::ios\\_base::unitbuf](#).

**uppercase()**

```
ios_base & std::uppercase (
    ios_base & __base) [inline]
```

Calls `base.setf(ios_base::uppercase)`.

References [std::ios\\_base::uppercase](#).

**ws()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::ws (
    basic_istream< _CharT, _Traits > & __is)
```

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass mc;

std::cin >> std::ws >> mc;
```



will skip leading whitespace before calling `operator>>` on `cin` and your object. Note that the same effect can be achieved by creating a `std::basic_istream::sentry` inside your definition of `operator>>`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [use\\_facet\(\)](#).

#### 4.10.5 Variable Documentation

##### **cerr**

`ostream` `std::cerr` [extern]

Linked to standard error (unbuffered)

##### **cin**

`istream` `std::cin` [extern]

Linked to standard input.

##### **clog**

`ostream` `std::clog` [extern]

Linked to standard error (buffered)

##### **cout**

`ostream` `std::cout` [extern]

Linked to standard output.

##### **default\_sentinel**

`default_sentinel_t` `std::default_sentinel` [inline], [constexpr]

A default sentinel value.

##### **destroying\_delete**

`destroying_delete_t` `std::destroying_delete` [inline], [constexpr]

Tag variable of type `destroying_delete_t`.

##### **ignore**

`_Swallow_assign` `std::ignore` [inline], [constexpr]

Used with `std:::tie` to ignore an element of a tuple

When using `std:::tie` to assign the elements of a tuple to variables, unwanted elements can be ignored by using `std:::ignore`. For example:

```
int x, y;  
std::tie(x, std::ignore, y) = std::make_tuple(1, 2, 3);
```

This assignment will perform `x=1; std::ignore=2; y=3;` which results in the second element being ignored.

Since

C++11

##### **wcerr**

`wostream` `std::wcerr` [extern]

Linked to standard error (unbuffered)

**wcin**

`wistream` `std::wcin` [extern]

Linked to standard input.

**wclog**

`wostream` `std::wclog` [extern]

Linked to standard error (buffered)

**wcout**

`wostream` `std::wcout` [extern]

Linked to standard output.

**4.11 std::\_\_debug Namespace Reference****Classes**

- class `bitset`
- class `deque`
- class `forward_list`
- class `list`
- class `map`
- class `multimap`
- class `multiset`
- class `set`
- class `unordered_map`
- class `unordered_multimap`
- class `unordered_multiset`
- class `unordered_set`
- class `vector`

**Functions**

- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**deque** (`_InputIterator`, `_InputIterator`, `_Allocator=_Allocator()`) -> `deque<_ValT, _Allocator>`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>`  
**deque** (`size_t`, `_Tp`, `_Allocator=_Allocator()`) -> `deque<_Tp, _Allocator>`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**forward\_list** (`_InputIterator`, `_InputIterator`, `_Allocator=_Allocator()`) -> `forward_list<_ValT, _Allocator>`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>`  
**forward\_list** (`size_t`, `_Tp`, `_Allocator=_Allocator()`) -> `forward_list<_Tp, _Allocator>`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**list** (`_InputIterator`, `_InputIterator`, `_Allocator=_Allocator()`) -> `list<_ValT, _Allocator>`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>`  
**list** (`size_t`, `_Tp`, `_Allocator=_Allocator()`) -> `list<_Tp, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**map** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, less<__iter_key_t<_InputIterator>>, _Allocator>`

- `template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_key_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Compare, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`map (initializer_list< pair< _Key, _Tp> >, _Allocator) -> map< _Key, _Tp, less< _Key>, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Compare = less< _Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`map (initializer_list< pair< _Key, _Tp> >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _Key, _Tp, _Compare, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, less< __iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_key_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< __iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Compare, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`multimap (initializer_list< pair< _Key, _Tp> >, _Allocator) -> multimap< _Key, _Tp, less< _Key>, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Compare = less< _Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`multimap (initializer_list< pair< _Key, _Tp> >, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator_traits<_InputIterator>::value_type, less< typename iterator_traits<_InputIterator>::value_type>, _Allocator>`
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator_traits<_InputIterator>::value_type, _Compare, _Allocator>`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`multiset (initializer_list< _Key>, _Allocator) -> multiset< _Key, less< _Key>, _Allocator>`
- `template<typename _Key, typename _Compare = less< _Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`multiset (initializer_list< _Key>, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _Compare, _Allocator>`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb> operator& (const bitset< _Nb> &__x, const bitset< _Nb> &__y) noexcept`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits> & operator<< (std::basic_ostream< _CharT, _Traits> &__os, const bitset< _Nb> &__x)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp> operator<=> (const deque< _Tp, _Alloc> &__x, const deque< _Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp> operator<=> (const forward_list< _Tp, _Alloc> &__x, const forward_list< _Tp, _Alloc> &__y)`

- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp > operator<=> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< pair< const _Key, _Tp > > operator<=> (const map< _Key, _Tp, _Compare, _Alloc > &__lhs, const map< _Key, _Tp, _Compare, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< pair< const _Key, _Tp > > operator<=> (const multimap< _Key, _Tp, _Compare, _Alloc > &__lhs, const multimap< _Key, _Tp, _Compare, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< _Key > operator<=> (const multiset< _Key, _Compare, _Alloc > &__lhs, const multiset< _Key, _Compare, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< _Key > operator<=> (const set< _Key, _Compare, _Alloc > &__lhs, const set< _Key, _Compare, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp > operator<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc>`  
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`

- `template<size_t _Nb>`  
`constexpr bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >`
- `template<typename _Tp, typename _Alloc>`  
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Tp, typename _Alloc>`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`
- `template<typename _Tp, typename _Alloc>`  
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Alloc>`  
`constexpr void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_map** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator >>, equal_to< __iter_key_t< _InputIterator >>, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_map** (`_InputIterator`, `_InputIterator`, `typename unordered_map< int, int >::size_type`, `_Allocator`) -> `unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator >>, equal_to< __iter_key_t< _InputIterator >>, _Allocator >`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_map** (`_InputIterator`, `_InputIterator`, `typename unordered_map< int, int >::size_type`, `_Hash`, `_Allocator`) -> `unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator >>, _Allocator >`
- `template<typename _InputIterator, typename _Hash = hash< __iter_key_t< _InputIterator >>, typename _Pred = equal_to< __iter_key_t< _InputIterator >>, typename _Allocator = allocator< __iter_to_alloc_t< _InputIterator >>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_map** (`_InputIterator`, `_InputIterator`, `typename unordered_map< int, int >::size_type={}`, `_Hash=Hash()`, `_Pred=_Pred()`, `_Allocator=_Allocator()`) -> `unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**unordered\_map** (`initializer_list< pair< _Key, _Tp >>`, `_Allocator`) -> `unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**unordered\_map** (`initializer_list< pair< _Key, _Tp >>`, `typename unordered_map< int, int >::size_type`, `_Allocator`) -> `unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_map** (`initializer_list< pair< _Key, _Tp >>`, `typename unordered_map< int, int >::size_type`, `_Hash`, `_Allocator`) -> `unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Hash = hash< _Key >, typename _Pred = equal_to< _Key >, typename _Allocator = allocator< pair< const _Key, _Tp >>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_map** (`initializer_list< pair< _Key, _Tp >>`, `typename unordered_map< int, int >::size_type={}`, `_Hash=Hash()`, `_Pred=_Pred()`, `_Allocator=_Allocator()`) -> `unordered_map< _Key, _Tp, _Hash, _Pred, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multimap** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator >>, equal_to< __iter_key_t< _InputIterator >>, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap< int, int >::size_type`, `_Allocator`) -> `unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator >>, equal_to< __iter_key_t< _InputIterator >>, _Allocator >`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap< int, int >::size_type`, `_Hash`, `_Allocator`) -> `unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator >>, _Allocator >`

- template<typename \_InputIterator, typename \_Hash = hash<\_\_iter\_key\_t<\_InputIterator>>, typename \_Pred = equal\_to<\_\_iter\_key\_t<\_InputIterator>>, typename \_Allocator = allocator<\_\_iter\_to\_alloc\_t<\_InputIterator>>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>>  
**unordered\_multimap** (\_InputIterator, \_InputIterator, unordered\_multimap< int, int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_multimap< \_\_iter\_key\_t<\_InputIterator>, \_\_iter\_val\_t<\_InputIterator>, \_Hash, \_Pred, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, \_Allocator) -> unordered\_multimap< \_Key, \_Tp, hash< \_Key >, equal\_to< \_Key >, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>  
**unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, unordered\_multimap< int, int >::size\_type, \_Allocator) -> unordered\_multimap< \_Key, \_Tp, hash< \_Key >, equal\_to< \_Key >, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Hash, typename \_Allocator, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>>  
**unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, unordered\_multimap< int, int >::size\_type, \_Hash, \_Allocator) -> unordered\_multimap< \_Key, \_Tp, \_Hash, equal\_to< \_Key >, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Hash = hash<\_Key>, typename \_Pred = equal\_to<\_Key>, typename \_Allocator = allocator<pair<const \_Key, \_Tp>>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>>  
**unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, unordered\_multimap< int, int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Allocator >
- template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>  
**unordered\_multiset** (\_InputIterator, \_InputIterator, \_Allocator) -> unordered\_multiset< typename iterator\_traits<\_InputIterator>::value\_type, hash< typename iterator\_traits<\_InputIterator>::value\_type >, equal\_to< typename iterator\_traits<\_InputIterator>::value\_type >, \_Allocator >
- template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>  
**unordered\_multiset** (\_InputIterator, \_InputIterator, unordered\_multiset< int >::size\_type, \_Allocator) -> unordered\_multiset< typename iterator\_traits<\_InputIterator>::value\_type, hash< typename iterator\_traits<\_InputIterator>::value\_type >, equal\_to< typename iterator\_traits<\_InputIterator>::value\_type >, \_Allocator >
- template<typename \_InputIterator, typename \_Hash, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>>  
**unordered\_multiset** (\_InputIterator, \_InputIterator, unordered\_multiset< int >::size\_type, \_Hash, \_Allocator) -> unordered\_multiset< typename iterator\_traits<\_InputIterator>::value\_type, \_Hash, equal\_to< typename iterator\_traits<\_InputIterator>::value\_type >, \_Allocator >
- template<typename \_InputIterator, typename \_Hash = hash<typename iterator\_traits<\_InputIterator>::value\_type>, typename \_Pred = equal\_to<typename iterator\_traits<\_InputIterator>::value\_type>, typename \_Allocator = allocator<typename iterator\_traits<\_InputIterator>::value\_type>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>>  
**unordered\_multiset** (\_InputIterator, \_InputIterator, unordered\_multiset< int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_multiset< typename iterator\_traits<\_InputIterator>::value\_type, \_Hash, \_Pred, \_Allocator >
- template<typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>>  
**unordered\_multiset** (initializer\_list< \_Tp >, \_Allocator) -> unordered\_multiset< \_Tp, hash< \_Tp >, equal\_to< \_Tp >, \_Allocator >
- template<typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>>  
**unordered\_multiset** (initializer\_list< \_Tp >, unordered\_multiset< int >::size\_type, \_Allocator) -> unordered\_multiset< \_Tp, hash< \_Tp >, equal\_to< \_Tp >, \_Allocator >
- template<typename \_Tp, typename \_Hash, typename \_Allocator, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>>



- unordered\_multiset** ([initializer\\_list](#)< \_Tp >, unordered\_multiset< int >::size\_type, \_Hash, \_Allocator) -> unordered\_multiset< \_Tp, \_Hash, [equal\\_to](#)< \_Tp >, \_Allocator >
- template<typename \_Tp, typename \_Hash = hash<\_Tp>, typename \_Pred = equal\_to<\_Tp>, typename \_Allocator = allocator<\_Tp>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>>
 **unordered\_multiset** ([initializer\\_list](#)< \_Tp >, unordered\_multiset< int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_multiset< \_Tp, \_Hash, \_Pred, \_Allocator >
  - template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>
**unordered\_set** (\_InputIterator, \_InputIterator, \_Allocator) -> unordered\_set< typename [iterator\\_traits](#)< \_InputIterator >::value\_type, hash< typename [iterator\\_traits](#)< \_InputIterator >::value\_type >, [equal\\_to](#)< typename [iterator\\_traits](#)< \_InputIterator >::value\_type >, \_Allocator >
  - template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>
**unordered\_set** (\_InputIterator, \_InputIterator, unordered\_set< int >::size\_type, \_Allocator) -> unordered\_set< typename [iterator\\_traits](#)< \_InputIterator >::value\_type, hash< typename [iterator\\_traits](#)< \_InputIterator >::value\_type >, [equal\\_to](#)< typename [iterator\\_traits](#)< \_InputIterator >::value\_type >, \_Allocator >
  - template<typename \_InputIterator, typename \_Hash, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>>
**unordered\_set** (\_InputIterator, \_InputIterator, unordered\_set< int >::size\_type, \_Hash, \_Allocator) -> unordered\_set< typename [iterator\\_traits](#)< \_InputIterator >::value\_type, \_Hash, [equal\\_to](#)< typename [iterator\\_traits](#)< \_InputIterator >::value\_type >, \_Allocator >
  - template<typename \_InputIterator, typename \_Hash = hash<typename [iterator\\_traits](#)<\_InputIterator>::value\_type>, typename \_Pred = equal\_to<typename [iterator\\_traits](#)<\_InputIterator>::value\_type>, typename \_Allocator = allocator<typename [iterator\\_traits](#)<\_InputIterator>::value\_type>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>>
**unordered\_set** (\_InputIterator, \_InputIterator, unordered\_set< int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_set< typename [iterator\\_traits](#)< \_InputIterator >::value\_type, \_Hash, \_Pred, \_Allocator >
  - template<typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>>
**unordered\_set** ([initializer\\_list](#)< \_Tp >, \_Allocator) -> unordered\_set< \_Tp, hash< \_Tp >, [equal\\_to](#)< \_Tp >, \_Allocator >
  - template<typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>>
**unordered\_set** ([initializer\\_list](#)< \_Tp >, unordered\_set< int >::size\_type, \_Allocator) -> unordered\_set< \_Tp, hash< \_Tp >, [equal\\_to](#)< \_Tp >, \_Allocator >
  - template<typename \_Tp, typename \_Hash, typename \_Allocator, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>>
**unordered\_set** ([initializer\\_list](#)< \_Tp >, unordered\_set< int >::size\_type, \_Hash, \_Allocator) -> unordered\_set< \_Tp, \_Hash, [equal\\_to](#)< \_Tp >, \_Allocator >
  - template<typename \_Tp, typename \_Hash = hash<\_Tp>, typename \_Pred = equal\_to<\_Tp>, typename \_Allocator = allocator<\_Tp>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>>
**unordered\_set** ([initializer\\_list](#)< \_Tp >, unordered\_set< int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_set< \_Tp, \_Hash, \_Pred, \_Allocator >
  - template<typename \_InputIterator, typename \_ValT = typename [iterator\\_traits](#)<\_InputIterator>::value\_type, typename \_Allocator = allocator<\_ValT>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>
**vector** (\_InputIterator, \_InputIterator, \_Allocator=\_Allocator()) -> vector< \_ValT, \_Allocator >
  - template<typename \_Tp, typename \_Allocator = allocator<\_Tp>, typename = \_RequireAllocator<\_Allocator>>>
**vector** (size\_t, \_Tp, \_Allocator=\_Allocator()) -> vector< \_Tp, \_Allocator >

#### 4.11.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior.



Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_xxx` macros are merely wrappers around the `__glibcxx_check_xxx` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

#### 4.11.2 Function Documentation

##### swap()

```
template<typename _Tp, typename _Alloc>
void std::__debug::swap (
    forward_list< _Tp, _Alloc > & __lx,
    forward_list< _Tp, _Alloc > & __ly) [inline], [noexcept]
```

See `std::forward_list::swap()`.

## 4.12 std::\_\_detail Namespace Reference

### Classes

- struct [\\_BracketMatcher](#)
- class [\\_Compiler](#)
- class [\\_Executor](#)
- struct [\\_List\\_node\\_base](#)
- struct [\\_List\\_node\\_header](#)
- struct [\\_Quoted\\_string](#)
- class [\\_Scanner](#)
- class [\\_StateSeq](#)

### Typedefs

- template<typename \_Tp, typename \_Up>  
using [\\_\\_cmp3way\\_res\\_t](#)
- template<typename \_Tp>  
using [\\_\\_unsigned\\_least\\_t](#)
- template<typename \_CharT>  
using [\\_Matcher](#)
- typedef long [\\_StateIdT](#)

### Enumerations

- enum [\\_Opcode](#) : int {  
    [\\_S\\_opcode\\_unknown](#) , [\\_S\\_opcode\\_alternative](#) , [\\_S\\_opcode\\_repeat](#) , [\\_S\\_opcode\\_backref](#) ,  
    [\\_S\\_opcode\\_line\\_begin\\_assertion](#) , [\\_S\\_opcode\\_line\\_end\\_assertion](#) , [\\_S\\_opcode\\_word\\_boundary](#) , [\\_S\\_opcode\\_subexpr\\_lookahead](#) ,  
    [\\_S\\_opcode\\_subexpr\\_begin](#) , [\\_S\\_opcode\\_subexpr\\_end](#) , [\\_S\\_opcode\\_dummy](#) , [\\_S\\_opcode\\_match](#) ,  
    [\\_S\\_opcode\\_accept](#) }
- enum class [\\_RegexExecutorPolicy](#) : int { [\\_S\\_auto](#) , [\\_S\\_alternate](#) }

### Functions

- template<typename \_Res, typename \_Tp>  
constexpr \_Res [\\_\\_abs\\_r](#) (\_Tp \_\_val)
- template<typename>  
void [\\_\\_abs\\_r](#) (bool)=delete

- `template<typename _ChTraits>`  
`constexpr auto __char_traits_cmp_cat (int __cmp) noexcept`
- `template<typename... _Ts>`  
`constexpr auto __common_cmp_cat ()`
- `template<typename _Container, typename _UnsafeContainer, typename _Predicate>`  
`_Container::size_type __erase_nodes_if (_Container &__cont, _UnsafeContainer &__ucont, _Predicate __pred)`
- `template<typename _ValT, typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & __extract_params (basic_istream< _CharT, _Traits > &__is, vector< _ValT > &__vals, size_t __n)`
- `template<bool _DecOnly, typename _Tp>`  
`constexpr bool __from_chars_alnum (const char * &__first, const char * __last, _Tp &__val, int __base)`
- `template<bool _DecOnly = false>`  
`constexpr unsigned char __from_chars_alnum_to_val (unsigned char __c)`
- `template<bool _DecOnly, typename _Tp>`  
`constexpr bool __from_chars_pow2_base (const char * &__first, const char * __last, _Tp &__val, int __base)`
- `template<typename _Tp>`  
`constexpr _Tp __gcd (_Tp __m, _Tp __n)`
- `template<typename _Tp>`  
`_Tp * __get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _Tp>`  
`constexpr bool __p1_representable_as_double (_Tp __x) noexcept`
- `template<typename _Tp>`  
`constexpr bool __raise_and_add (_Tp &__val, int __base, unsigned char __c)`
- `template<typename _Bilter, typename _Alloc, typename _CharT, typename _TraitsT>`  
`bool __regex_algo_impl (_Bilter __s, _Bilter __e, match_results< _Bilter, _Alloc > &__m, const basic_regex< _CharT, _TraitsT > &__re, regex_constants::match_flag_type __flags, _RegexExecutorPolicy __policy, bool __match_mode)`
- `template<typename _Tp>`  
`constexpr bool __representable_as_double (_Tp __x) noexcept`
- `template<typename _Tp>`  
`void __return_temporary_buffer (_Tp * __p, size_t __len)`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars (char * __first, char * __last, _Tp __val, int __base) noexcept`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars_10 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr void __to_chars_10_impl (char * __first, unsigned __len, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars_16 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars_2 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars_8 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr unsigned __to_chars_len (_Tp __value, int __base) noexcept`
- `template<typename _Tp>`  
`constexpr unsigned __to_chars_len_2 (_Tp __value) noexcept`
- `template<typename _CharT, typename _Traits, typename _String>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Quoted_string< _String, _CharT > &__str)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Quoted_string< const _CharT *, _CharT > &__str)`

- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, const`  
`_Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > &__str)`

## Variables

- `template<typename _Tp>`  
`constexpr unsigned __cmp_cat_id`
- `template<> constexpr unsigned __cmp_cat_id< partial_ordering >`
- `template<> constexpr unsigned __cmp_cat_id< strong_ordering >`
- `template<> constexpr unsigned __cmp_cat_id< weak_ordering >`
- `constexpr _StateIdT _S_invalid_state_id`

### 4.12.1 Detailed Description

Implementation details not part of the namespace std interface.

### 4.12.2 Function Documentation

#### `__from_chars_alnum()`

```
template<bool _DecOnly, typename _Tp>
bool std::__detail::__from_chars_alnum (
    const char *& __first,
    const char * __last,
    _Tp & __val,
    int __base) [constexpr]
```

`std::from_chars` implementation for integers in any base. If `_DecOnly` is true, then we may assume `__base` is at most 10.

#### `__from_chars_pow2_base()`

```
template<bool _DecOnly, typename _Tp>
bool std::__detail::__from_chars_pow2_base (
    const char *& __first,
    const char * __last,
    _Tp & __val,
    int __base) [constexpr]
```

`std::from_chars` implementation for integers in a power-of-two base. If `_DecOnly` is true, then we may assume `__base` is at most 8.

#### `operator<<()` [1/2]

```
template<typename _CharT, typename _Traits, typename _String>
std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const _Quoted_string< _String, _CharT > & __str)
```

Insertion for quoted strings.

#### `operator<<()` [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const _Quoted_string< const _CharT *, _CharT > & __str)
```

Insertion for quoted strings.

**operator>>()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits > & std::__detail::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > & __str)
```

Extractor for delimited strings. The left and right delimiters can be different.

References [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::fmtflags](#), [std::basic\\_ios< \\_CharT, \\_Traits >::get\(\)](#), [std::ios\\_base::setf\(\)](#), [std::ios\\_base::skipws](#), and [std::basic\\_istream< \\_CharT, \\_Traits >::unset\(\)](#).

**4.13 std::\_\_parallel Namespace Reference****Classes**

- struct [\\_CRandNumber](#)

**Functions**

- template<typename \_\_RAIter, typename \_Tp, typename \_BinaryOperation>  
\_Tp \_\_**accumulate\_switch** (\_\_RAIter \_\_begin, \_\_RAIter \_\_end, \_Tp \_\_init, \_BinaryOperation \_\_binary\_op, [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism\_tag)
- template<typename \_Iter, typename \_Tp, typename \_BinaryOperation, typename \_IteratorTag>  
\_Tp \_\_**accumulate\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_Tp \_\_init, \_BinaryOperation \_\_binary\_op, \_IteratorTag)
- template<typename \_Iter, typename \_Tp, typename \_IteratorTag>  
\_Tp \_\_**accumulate\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_Tp \_\_init, \_IteratorTag)
- template<typename \_Iter, typename \_Tp, typename \_BinaryOper, typename \_Tag>  
\_Tp \_\_**accumulate\_switch** (\_Iter, \_Iter, \_Tp, \_BinaryOper, \_Tag)
- template<typename \_Iter, typename \_Tp, typename \_Tag>  
\_Tp \_\_**accumulate\_switch** (\_Iter, \_Iter, \_Tp, \_Tag)
- template<typename \_RAIter, typename \_Tp, typename \_BinaryOper>  
\_Tp \_\_**accumulate\_switch** (\_RAIter, \_RAIter, \_Tp, \_BinaryOper, [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism=[\\_\\_gnu\\_parallel::parallel\\_unbalanced](#))
- template<typename \_Iter, typename \_OutputIterator, typename \_BinaryOperation, typename \_IteratorTag1, typename \_IteratorTag2>  
\_OutputIterator \_\_**adjacent\_difference\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result, \_BinaryOperation \_\_bin\_op, \_IteratorTag1, \_IteratorTag2)
- template<typename \_Iter, typename \_OutputIterator, typename \_BinaryOperation>  
\_OutputIterator \_\_**adjacent\_difference\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result, \_BinaryOperation \_\_bin\_op, [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism\_tag)
- template<typename \_Iter, typename \_OIter, typename \_BinaryOper, typename \_Tag1, typename \_Tag2>  
\_OIter \_\_**adjacent\_difference\_switch** (\_Iter, \_Iter, \_OIter, \_BinaryOper, \_Tag1, \_Tag2)
- template<typename \_Iter, typename \_OIter, typename \_BinaryOper>  
\_OIter \_\_**adjacent\_difference\_switch** (\_Iter, \_Iter, \_OIter, \_BinaryOper, [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism=[\\_\\_gnu\\_parallel::parallel\\_unbalanced](#))
- template<typename \_Filter, typename \_BiPredicate, typename \_IterTag>  
\_Filter \_\_**adjacent\_find\_switch** (\_Filter, \_Filter, \_BiPredicate, \_IterTag)
- template<typename \_Filter, typename \_IterTag>  
\_Filter \_\_**adjacent\_find\_switch** (\_Filter, \_Filter, \_IterTag)
- template<typename \_Filterator, typename \_BinaryPredicate, typename \_IteratorTag>  
\_Filterator \_\_**adjacent\_find\_switch** (\_Filterator \_\_begin, \_Filterator \_\_end, \_BinaryPredicate \_\_pred, \_IteratorTag)
- template<typename \_Filterator, typename \_IteratorTag>  
\_Filterator \_\_**adjacent\_find\_switch** (\_Filterator \_\_begin, \_Filterator \_\_end, \_IteratorTag)

- `template<typename _RAIter, typename _BinaryPredicate>`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random\_access\_iterator\_tag)`
- `template<typename _RAIter>`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random\_access\_iterator\_tag)`
- `template<typename _RAIter, typename _BiPredicate>`  
`_RAIter __adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag>`  
`iterator\_traits< _Iter >::difference_type __count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag>`  
`iterator\_traits< _Iter >::difference_type __count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`iterator\_traits< _RAIter >::difference_type __count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`iterator\_traits< _Iter >::difference_type __count_switch (_Iter __begin, _Iter __end, const _Tp &__value, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag>`  
`iterator\_traits< _Iter >::difference_type __count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp>`  
`iterator\_traits< _RAIter >::difference_type __count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`bool __equal_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`bool __equal_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Filterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Filterator, typename _IteratorTag1, typename _IteratorTag2>`  
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2>`  
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2>`  
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filterator, typename _BinaryPredicate, typename _IteratorTag>`  
`_RAIter __find_first_of_switch (_RAIter __begin1, _RAIter __end1, _Filterator __begin2, _Filterator __end2, _BinaryPredicate __comp, random\_access\_iterator\_tag, _IteratorTag)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag>`  
`_RAIter __find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag>`  
`_Iter __find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag>`  
`_Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter __find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`_Iter __find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`

- `template<typename _Iter, typename _Tp, typename _IterTag>`  
`_Iter find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp>`  
`_RAIter find_switch (_RAIter __begin, _RAIter __end, const _Tp & __val, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag>`  
`_Function for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _Iter, typename _Function, typename _IterTag>`  
`_Function for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _RAIter, typename _Function>`  
`_Function for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag>`  
`_OIter generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag>`  
`_OutputIterator generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator>`  
`_RAIter generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator, typename _IterTag>`  
`void generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _Filterator, typename _Generator, typename _IteratorTag>`  
`void generate_switch (_Filterator __begin, _Filterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator>`  
`void generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IterTag1, typename _IterTag2>`  
`_Tp inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IterTag1, _IterTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2>`  
`_Tp inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag, gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`bool lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`bool lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`bool lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag>`  
`_Filter max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag>`  
`_Filterator max_element_switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename`  
`_IterTag3>`  
`_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag>`  
`_Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag>`  
`_Filterator __min_element_switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag,  
gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,  
_Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate ↵`  
`__pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,  
_Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,  
_RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation`  
`__bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation`  
`__bin_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _IterTag>`  
`_Filter __partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _IteratorTag>`  
`_Filterator __partition_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag>`  
`void __replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp, typename _IteratorTag>`  
`void __replace_if_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp & __new_value,  
_IteratorTag)`

- `template<typename _RAIter, typename _Predicate, typename _Tp>`  
`void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value,`  
`random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag>`  
`void __replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag>`  
`void __replace_switch (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Tp>`  
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value,`  
`random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag>`  
`_Filter __search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag>`  
`_FIterator __search_n_switch (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, __↵`  
`BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _Binary↵`  
`Predicate __binary_pred, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2>`  
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2>`  
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __↵`  
`end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2>`  
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __↵`  
`end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate>`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, __↵`  
`BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2>`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2,`  
`random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate>`  
`_RAIter1 __search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag,`  
`random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename __↵`  
`IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator __set_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,`  
`_OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename`  
`_IterTag3>`  
`_OIter __set_difference_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _Iter↵`  
`Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter __set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,`  
`random_access_iterator_tag)`



- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator __set_intersection_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter __set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator __set_symmetric_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter __set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter __set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator __set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter __set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter __set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2>`  
`_OIter __transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation>`  
`_RAOIter __transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation>`  
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3>`  
`_OutputIterator __transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3>`  
`_OIter __transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`

- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation>`  
`_RAIter3 __transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, ↵`  
`_BinaryOperation __binary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
\_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation>`  
`_RAIter3 __transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism= \_\_gnu\_parallel::parallel\_ba`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator __unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred,  
_IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`_OIter __unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccessOutputIterator, typename _Predicate>`  
`_RandomAccessOutputIterator __unique_copy_switch (_RAIter __begin, _RAIter __last, _RandomAccess↵`  
`OutputIterator __out, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate>`  
`_RandomAccess_OIter __unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, \_\_gnu\_parallel::\_\_Parallelism  
\_\_parallelism\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Tp accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp>`  
`_Tp accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter, typename _Tp>`  
`_Tp accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::\_\_Parallelism  
\_\_parallelism\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation  
__bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation  
__binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation  
__binary_op, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _OIter>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter, typename _OIter>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter adjacent_find (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate>`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate>`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator>`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator>`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate>`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate>`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp>`  
`iterator\_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Tp>`  
`iterator\_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp>`  
`iterator\_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator\_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`iterator\_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator\_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __↵`  
`binary_pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp>`  
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator>`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIterator>`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate>`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate`  
`__comp)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate>`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate`  
`__comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter>`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Filter>`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate>`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate>`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate>`  
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Function>`  
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Function>`  
`_Function for_each (_Iter, _Iter, _Function)`
- `template<typename _Iterator, typename _Function>`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iterator, typename _Function>`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __↵`  
`parallelism_tag)`
- `template<typename _Filter, typename _Generator>`  
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator>`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Generator>`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator>`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`

- `template<typename _FIterator, typename _Generator>`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism __↔  
parallelism_tag)`
- `template<typename _FIterator, typename _Generator>`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism  
__parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Tp>`  
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp)`
- `template<typename _IIter1, typename _IIter2, typename _Tp>`  
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _IIter1, typename _IIter2, typename _Tp>`  
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp inner_product (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2>`  
`constexpr bool lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)`
- `template<typename _IIter1, typename _IIter2>`  
`bool lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,  
\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate>`  
`constexpr bool lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,  
_Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate>`  
`bool lexicographical_compare (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate  
__pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter max_element (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter>`  
`_Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare>`  
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare>`  
`_Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare>`  
`_Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _FIterator>`  
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator>`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _FIterator>`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare>`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _FIterator, typename _Compare>`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare>`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵ result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵ result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵ result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵ result, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter min_element (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Filter>`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare>`  
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare>`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Filter, typename _Compare>`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator>`  
`_FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator>`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _FIterator>`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare>`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp)`

- `template<typename _FIterator, typename _Compare>`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism`  
`__parallelism_tag)`
- `template<typename _FIterator, typename _Compare>`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred,  
\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __begin1, _InputIterator1 __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter>`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _RAIter>`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OIter>`  
`_OIter partial_sum (_Iter, _Iter, _OIter __result)`



- `template<typename _Iter, typename _OIter>`  
`_OIter partial_sum (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate>`  
`_Filter partition (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Predicate>`  
`_Filter partition (_Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Predicate>`  
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred)`
- `template<typename _Filterator, typename _Predicate>`  
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter>`  
`void random_shuffle (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter>`  
`void random_shuffle (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _RandomNumberGenerator>`  
`void random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _RAlter, typename _RandomNumberGenerator>`  
`void random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp>`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp>`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Tp>`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Tp>`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Filterator, typename _Tp>`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Tp>`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp>`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filterator, typename _Predicate, typename _Tp>`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp>`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2>`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`



- `template<typename _Filter1, typename _Filter2>`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate>`  
`constexpr _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate>`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2>`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2>`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>`  
`constexpr _FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`_ForwardIterator search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`
- `template<typename _Filter, typename _Integer, typename _Tp>`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp>`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp>`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp>`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator>`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator>`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter>`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter>`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`

- [illegible]

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism>`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _RAIter, typename _Compare, typename _Parallelism>`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op,`  
`\_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::Parallelism`  
`BinaryOperation __binary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::Parallelism`  
`BinaryOperation __binary_op, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag`  
`BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter>`  
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter>`  
`_OIter unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`

#### 4.13.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

### 4.13.2 Function Documentation

#### search()

```
template<typename _ForwardIterator, typename _Searcher>
_FowardIterator std::__parallel::search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Searcher & __searcher) [inline]
```

Search a sequence using a Searcher object.

#### Parameters

|                         |                     |
|-------------------------|---------------------|
| <code>__first</code>    | A forward iterator. |
| <code>__last</code>     | A forward iterator. |
| <code>__searcher</code> | A callable object.  |

#### Returns

```
__searcher(__first, __last).first
```

## 4.14 std::chrono Namespace Reference

### Classes

- class [duration](#)
- struct [duration\\_values](#)
- class [gps\\_clock](#)
- class [hh\\_mm\\_ss](#)
- struct [steady\\_clock](#)
- struct [system\\_clock](#)
- class [tai\\_clock](#)
- class [time\\_point](#)
- struct [treat\\_as\\_floating\\_point](#)
- class [tzdb\\_list](#)
- class [utc\\_clock](#)

### Typedefs

- using [days](#)
- using [file\\_clock](#)
- template<typename \_Duration>  
using [file\\_time](#)
- using [gps\\_seconds](#)
- template<typename \_Duration>  
using [gps\\_time](#)
- using [high\\_resolution\\_clock](#)
- using [hours](#)
- using [local\\_days](#)
- using [local\\_seconds](#)
- template<typename \_Duration>  
using [local\\_time](#)
- using [microseconds](#)
- using [milliseconds](#)

- using [minutes](#)
- using [months](#)
- using [nanoseconds](#)
- using [seconds](#)
- using **sys\_days**
- using **sys\_seconds**
- template<typename \_Duration>  
using **sys\_time**
- using **tai\_seconds**
- template<typename \_Duration>  
using **tai\_time**
- using **utc\_seconds**
- template<typename \_Duration>  
using **utc\_time**
- using [weeks](#)
- using [years](#)
- using **zoned\_seconds**

## Enumerations

- enum class **choose** { **earliest** , **latest** }

## Functions

- template<typename \_Duration>  
void **\_\_throw\_bad\_local\_time** (const [local\\_time](#)< \_Duration > &\_\_tp, const local\_info &\_\_i)
- template<typename \_Rep, typename \_Period>  
constexpr [enable\\_if\\_t](#)< [numeric\\_limits](#)< \_Rep >::is\_signed, [duration](#)< \_Rep, \_Period > > [abs](#) ([duration](#)< \_Rep, \_Period > \_\_d)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr [\\_\\_enable\\_if\\_is\\_duration](#)< \_ToDur > [ceil](#) (const [duration](#)< \_Rep, \_Period > &\_\_d)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr [enable\\_if\\_t](#)< [\\_\\_is\\_duration\\_v](#)< \_ToDur >, [time\\_point](#)< \_Clock, \_ToDur > > [ceil](#) (const [time\\_point](#)< \_Clock, \_Dur > &\_\_tp)
- template<typename \_DestClock, typename \_SourceClock, typename \_Duration>  
requires [\\_\\_detail::\\_\\_clock\\_convs](#)< \_DestClock, \_SourceClock, \_Duration > || [\\_\\_detail::\\_\\_clock\\_convs\\_sys](#)< \_DestClock, \_SourceClock, [\\_\\_duration](#)> || [\\_\\_detail::\\_\\_clock\\_convs\\_utc](#)< \_DestClock, \_SourceClock, \_Duration > || [\\_\\_detail::\\_\\_clock\\_convs\\_sys\\_utc](#)< \_DestClock, [\\_\\_SourceClock](#), \_Duration > || [\\_\\_detail::\\_\\_clock\\_convs\\_utc\\_sys](#)< \_DestClock, \_SourceClock, \_Duration >  
auto [clock\\_cast](#) (const [time\\_point](#)< \_SourceClock, \_Duration > &\_\_t)
- const time\_zone \* **current\_zone** ()
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr [\\_\\_enable\\_if\\_is\\_duration](#)< \_ToDur > [duration\\_cast](#) (const [duration](#)< \_Rep, \_Period > &\_\_d)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr [\\_\\_enable\\_if\\_is\\_duration](#)< \_ToDur > [floor](#) (const [duration](#)< \_Rep, \_Period > &\_\_d)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr [enable\\_if\\_t](#)< [\\_\\_is\\_duration\\_v](#)< \_ToDur >, [time\\_point](#)< \_Clock, \_ToDur > > [floor](#) (const [time\\_point](#)< \_Clock, \_Dur > &\_\_tp)
- template<typename \_CharT, typename \_Traits, typename \_Alloc = allocator<\_CharT>>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, day &\_\_d, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Rep, typename \_Period, typename \_Alloc = allocator<\_CharT>>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, [duration](#)< \_Rep, \_Period > &\_\_d, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)

- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, file_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, gps_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, local_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, month &__m, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, month_day &__md, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, sys_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, tai_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, utc_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, weekday &__wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, year &__y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, year_month &__ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, year_month_day &__ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _Duration>`  
`leap_second_info get_leap_second_info (const utc_time< _Duration > &__ut)`
- `const tzdb & get_tzdb ()`
- `tzdb_list & get_tzdb_list ()`
- `constexpr bool is_am (const hours &__h) noexcept`
- `constexpr bool is_pm (const hours &__h) noexcept`
- `template<typename _Duration>`  
`__detail::__local_time_fmt< _Duration > local_time_format (local_time< _Duration > __time, const string *__`  
`__abbrev=nullptr, const seconds *__offset_sec=nullptr)`
- `const time_zone * locate_zone (string_view __tz_name)`

- constexpr [hours](#) **make12** (const [hours](#) &\_\_h) noexcept
- constexpr [hours](#) **make24** (const [hours](#) &\_\_h, bool \_\_is\_pm) noexcept
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const day &\_\_d)
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [file\\_time](#)< \_Duration > &\_\_t)
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [gps\\_time](#)< \_Duration > &\_\_t)
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [hh\\_mm\\_ss](#)< \_Duration > &\_\_hms)
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const local\_info &\_\_li)
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
requires requires(const [sys\\_time](#)<\_Duration>& \_\_st) { \_\_os << \_\_st; }  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [local\\_time](#)< \_Duration > &\_\_lt)
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const month &\_\_m)
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const month\_day &\_\_md)
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const month\_day\_last &\_\_mdl)
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const month\_weekday &\_\_mwd)
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const month\_weekday\_last &\_\_mwdl)
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [sys\\_days](#) &\_\_dp)
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const sys\_info &\_\_i)
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
requires (!treat\_as\_floating\_point\_v<typename \_Duration::rep>) && ratio\_less\_v<typename \_Duration::period, days::period>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [sys\\_time](#)< \_Duration > &\_\_tp)
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [tai\\_time](#)< \_Duration > &\_\_t)
- template<typename \_CharT, typename \_Traits, typename \_Duration>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [utc\\_time](#)< \_Duration > &\_\_t)
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const weekday &\_\_wd)



- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const weekday<↵`  
`_indexed & __wdi)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const weekday<↵`  
`_last & __wdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const year & __y)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const year_month`  
`& __ym)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const year<↵`  
`month_day & __ymd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const year<↵`  
`month_day_last & __ymdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const year<↵`  
`month_weekday & __ymwd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const year<↵`  
`month_weekday_last & __ymwdl)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const zoned<↵`  
`time< _Duration, _TimeZonePtr > & __t)`
- `template<typename _CharT, typename _Traits, typename _Rep, typename _Period>`  
`basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`duration< _Rep, _Period > & __d)`
- `template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>`  
`bool operator== (const zoned_time< _Dur1, _TZPtr1 > & __x, const zoned_time< _Dur2, _TZPtr2 > & __y)`
- `template<typename _CharT, __detail::__parsable< _CharT > _Parsable>`  
`auto parse (const _CharT * __fmt, _Parsable & __tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT > _Parsable>`  
`auto parse (const _CharT * __fmt, _Parsable & __tp, basic_string< _CharT, _Traits, _Alloc > & __abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto parse (const _CharT * __fmt, _Parsable & __tp, basic_string< _CharT, _Traits, _Alloc > & __abbrev, minutes`  
`& __offset)`
- `template<typename _CharT, typename _Traits = char_traits< _CharT>, typename _StrT = basic_string< _CharT, _Traits>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto parse (const _CharT * __fmt, _Parsable & __tp, minutes & __offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>`  
`auto parse (const basic_string< _CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT > _Parsable>`  
`auto parse (const basic_string< _CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp, basic_string< _CharT, ↵`  
`_Traits, _Alloc > & __abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto parse (const basic_string< _CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp, basic_string< _CharT, ↵`  
`_Traits, _Alloc > & __abbrev, minutes & __offset)`

- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable<_CharT, _Traits, _StrT, minutes> > _Parsable>`  
`auto parse (const basic_string<_CharT, _Traits, _Alloc> &__fmt, _Parsable &__tp, minutes &__offset)`
- `const tzdb & reload_tzdb ()`
- `string remote_version ()`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr enable_if_t<__and<__is_duration<_ToDur>, __not< treat_as_floating_point< typename _ToDur::rep> >>::value, _ToDur> round (const duration<_Rep, _Period> &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr enable_if_t<__is_duration_v<_ToDur> &&! treat_as_floating_point_v< typename _ToDur::rep>, time_point<_Clock, _ToDur> > round (const time_point<_Clock, _Dur> &__tp)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr __enable_if_t<__is_duration<_ToDur>::value, time_point<_Clock, _ToDur> > time_point_cast (const time_point<_Clock, _Dur> &__t)`
- `zoned_time () -> zoned_time< seconds>`
- `template<typename _TimeZonePtrOrName>`  
`zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation<_TimeZonePtrOrName> >`
- `template<typename _TimeZonePtrOrName, typename _Duration>`  
`zoned_time (_TimeZonePtrOrName &&, local_time<_Duration>, choose=choose::earliest) -> zoned_time< common_type_t<_Duration, seconds>, __time_zone_representation<_TimeZonePtrOrName> >`
- `template<typename _TimeZonePtrOrName, typename _Duration>`  
`zoned_time (_TimeZonePtrOrName &&, sys_time<_Duration>) -> zoned_time< common_type_t<_Duration, seconds>, __time_zone_representation<_TimeZonePtrOrName> >`
- `template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>`  
`zoned_time (_TimeZonePtrOrName &&, zoned_time<_Duration, _TimeZonePtr2>, choose=choose::earliest) -> zoned_time< common_type_t<_Duration, seconds>, __time_zone_representation<_TimeZonePtrOrName> >`
- `template<typename _Duration>`  
`zoned_time (sys_time<_Duration>) -> zoned_time< common_type_t<_Duration, seconds> >`
- `template<typename _Clock, typename _Dur1, three_way_comparable_with<_Dur1> _Dur2>`  
`constexpr auto operator<=> (const time_point<_Clock, _Dur1> &__lhs, const time_point<_Clock, _Dur2> &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool operator< (const time_point<_Clock, _Dur1> &__lhs, const time_point<_Clock, _Dur2> &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool operator<= (const time_point<_Clock, _Dur1> &__lhs, const time_point<_Clock, _Dur2> &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool operator> (const time_point<_Clock, _Dur1> &__lhs, const time_point<_Clock, _Dur2> &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool operator>= (const time_point<_Clock, _Dur1> &__lhs, const time_point<_Clock, _Dur2> &__rhs)`

## Variables

- `template<typename _Tp, typename _Clock>`  
`constexpr bool __is_time_point_for_v`
- `template<typename _Clock, typename _Duration>`  
`constexpr bool __is_time_point_for_v< time_point<_Clock, _Duration>, _Clock>`

- constexpr month **April**
- constexpr month **August**
- constexpr month **December**
- constexpr month **February**
- constexpr weekday **Friday**
- template<typename \_Tp, typename = void>  
constexpr bool **is\_clock\_v**
- template<typename \_Tp>  
constexpr bool **is\_clock\_v**< \_Tp, void\_t< typename \_Tp::rep, typename \_Tp::period, typename \_Tp::duration, typename \_Tp::time\_point::duration, decltype(\_Tp::is\_steady), decltype(\_Tp::now())> >
- template<> constexpr bool **is\_clock\_v**< **file\_clock** >
- template<> constexpr bool **is\_clock\_v**< **gps\_clock** >
- template<> constexpr bool **is\_clock\_v**< **steady\_clock** >
- template<> constexpr bool **is\_clock\_v**< **system\_clock** >
- template<> constexpr bool **is\_clock\_v**< **tai\_clock** >
- template<> constexpr bool **is\_clock\_v**< **utc\_clock** >
- constexpr month **January**
- constexpr month **July**
- constexpr month **June**
- constexpr last\_spec **last**
- constexpr month **March**
- constexpr month **May**
- constexpr weekday **Monday**
- constexpr month **November**
- constexpr month **October**
- constexpr weekday **Saturday**
- constexpr month **September**
- constexpr weekday **Sunday**
- constexpr weekday **Thursday**
- template<typename \_Rep>  
constexpr bool **treat\_as\_floating\_point\_v**
- template<> constexpr bool **treat\_as\_floating\_point\_v**< **double** >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< **float** >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< **int** >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< **long** >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< **long double** >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< **long long** >
- constexpr weekday **Tuesday**
- constexpr weekday **Wednesday**
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr **common\_type**< **duration**< \_Rep1, \_Period1 >, **duration**< \_Rep2, \_Period2 > >::type **operator-** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr **common\_type**< **duration**< \_Rep1, \_Period1 >, **duration**< \_Rep2, \_Period2 > >::type **operator+** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Rep2, typename \_Period>  
constexpr **duration**< \_\_common\_rep\_t< \_Rep2, \_Rep1 >, \_Period > **operator\*** (const \_Rep1 &\_\_s, const **duration**< \_Rep2, \_Period > &\_\_d)
- template<typename \_Rep1, typename \_Period, typename \_Rep2>  
constexpr **duration**< \_\_common\_rep\_t< \_Rep1, \_\_disable\_if\_is\_duration< \_Rep2 >, \_Period > > **operator/** (const **duration**< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)

- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr [common\\_type](#)< \_Rep1, \_Rep2 >::type [operator/](#) (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2>  
constexpr [duration](#)< \_\_common\_rep\_t< \_Rep1, \_\_disable\_if\_is\_duration< \_Rep2 >, \_Period > [operator%](#) (const [duration](#)< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr [common\\_type](#)< [duration](#)< \_Rep1, \_Period1 >, [duration](#)< \_Rep2, \_Period2 > >::type [operator%](#) (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2>  
constexpr [duration](#)< \_\_common\_rep\_t< \_Rep1, \_Rep2 >, \_Period > [operator\\*](#) (const [duration](#)< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool [operator<](#) (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
requires three\_way\_comparable<[common\\_type\\_t](#)<\_Rep1, \_Rep2>>  
constexpr auto [operator<=>](#) (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool [operator<=](#) (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool [operator>](#) (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool [operator>=](#) (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool [operator==](#) (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Clock, typename \_Dur2>  
constexpr [time\\_point](#)< \_Clock, typename [common\\_type](#)< [duration](#)< \_Rep1, \_Period1 >, \_Dur2 >::type > [operator+](#) (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [time\\_point](#)< \_Clock, \_Dur2 > &\_\_rhs)
- template<typename \_Clock, typename \_Dur1, typename \_Rep2, typename \_Period2>  
constexpr [time\\_point](#)< \_Clock, typename [common\\_type](#)< \_Dur1, [duration](#)< \_Rep2, \_Period2 > >::type > [operator-](#) (const [time\\_point](#)< \_Clock, \_Dur1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Clock, typename \_Dur1, typename \_Dur2>  
constexpr [common\\_type](#)< \_Dur1, \_Dur2 >::type [operator-](#) (const [time\\_point](#)< \_Clock, \_Dur1 > &\_\_lhs, const [time\\_point](#)< \_Clock, \_Dur2 > &\_\_rhs)
- template<typename \_Clock, typename \_Dur1, typename \_Rep2, typename \_Period2>  
constexpr [time\\_point](#)< \_Clock, typename [common\\_type](#)< \_Dur1, [duration](#)< \_Rep2, \_Period2 > >::type > [operator+](#) (const [time\\_point](#)< \_Clock, \_Dur1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)

#### 4.14.1 Detailed Description

ISO C++ 2011 namespace for date and time utilities.

## 4.15 std::decimal Namespace Reference

### Classes

- class [decimal128](#)

- class [decimal32](#)
- class [decimal64](#)

## Functions

- double **decimal128\_to\_double** ([decimal128](#) \_\_d)
- float **decimal128\_to\_float** ([decimal128](#) \_\_d)
- long double **decimal128\_to\_long\_double** ([decimal128](#) \_\_d)
- long long **decimal128\_to\_long\_long** ([decimal128](#) \_\_d)
- double **decimal32\_to\_double** ([decimal32](#) \_\_d)
- float **decimal32\_to\_float** ([decimal32](#) \_\_d)
- long double **decimal32\_to\_long\_double** ([decimal32](#) \_\_d)
- long long **decimal32\_to\_long\_long** ([decimal32](#) \_\_d)
- double **decimal64\_to\_double** ([decimal64](#) \_\_d)
- float **decimal64\_to\_float** ([decimal64](#) \_\_d)
- long double **decimal64\_to\_long\_double** ([decimal64](#) \_\_d)
- long long **decimal64\_to\_long\_long** ([decimal64](#) \_\_d)
- double **decimal\_to\_double** ([decimal128](#) \_\_d)
- double **decimal\_to\_double** ([decimal32](#) \_\_d)
- double **decimal\_to\_double** ([decimal64](#) \_\_d)
- float **decimal\_to\_float** ([decimal128](#) \_\_d)
- float **decimal\_to\_float** ([decimal32](#) \_\_d)
- float **decimal\_to\_float** ([decimal64](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal128](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal32](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal64](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal128](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal32](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal64](#) \_\_d)
- static [decimal128](#) **make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static [decimal128](#) **make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)

- bool **operator!=** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **operator\*** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **operator\*** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, int \_\_rhs)

- **decimal64 operator\*** (decimal64 \_\_lhs, long \_\_rhs)
- **decimal64 operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- **decimal64 operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator\*** (int \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator\*** (long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator\*** (long long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator\*** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator\*** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator\*** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator\*** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator\*** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator\*** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, int \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, long long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_rhs)
- **decimal128 operator+** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator+** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator+** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- **decimal32 operator+** (decimal32 \_\_lhs, int \_\_rhs)
- **decimal32 operator+** (decimal32 \_\_lhs, long \_\_rhs)
- **decimal32 operator+** (decimal32 \_\_lhs, long long \_\_rhs)
- **decimal32 operator+** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator+** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator+** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- **decimal32 operator+** (decimal32 \_\_rhs)
- **decimal128 operator+** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, int \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, long \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, long long \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, unsigned long \_\_rhs)



- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, long long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, int \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, long \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, int \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, long \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, long long \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_rhs)
- [decimal128 operator-](#) (int \_\_lhs, [decimal128](#) \_\_rhs)



- **decimal32 operator-** (int \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator-** (int \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator-** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator-** (long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator-** (long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator-** (long long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator-** (long long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator-** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator-** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator-** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator-** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator-** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator-** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, int \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator/** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, int \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, long \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, int \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, long \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** (int \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator/** (int \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator/** (int \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator/** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator/** (long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator/** (long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator/** (long long \_\_lhs, decimal128 \_\_rhs)

- [decimal32 operator/](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator/](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator/](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator/](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator/](#) (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator/](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator/](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator/](#) (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator/](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator/](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator/](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [bool operator<](#) ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [bool operator<](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [bool operator<](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [bool operator<](#) ([decimal128](#) \_\_lhs, int \_\_rhs)
- [bool operator<](#) ([decimal128](#) \_\_lhs, long \_\_rhs)
- [bool operator<](#) ([decimal128](#) \_\_lhs, long long \_\_rhs)
- [bool operator<](#) ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- [bool operator<](#) ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- [bool operator<](#) ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- [bool operator<](#) ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [bool operator<](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [bool operator<](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [bool operator<](#) ([decimal32](#) \_\_lhs, int \_\_rhs)
- [bool operator<](#) ([decimal32](#) \_\_lhs, long \_\_rhs)
- [bool operator<](#) ([decimal32](#) \_\_lhs, long long \_\_rhs)
- [bool operator<](#) ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [bool operator<](#) ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- [bool operator<](#) ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- [bool operator<](#) ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [bool operator<](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [bool operator<](#) ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [bool operator<](#) ([decimal64](#) \_\_lhs, int \_\_rhs)
- [bool operator<](#) ([decimal64](#) \_\_lhs, long \_\_rhs)
- [bool operator<](#) ([decimal64](#) \_\_lhs, long long \_\_rhs)
- [bool operator<](#) ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- [bool operator<](#) ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- [bool operator<](#) ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- [bool operator<](#) (int \_\_lhs, [decimal128](#) \_\_rhs)
- [bool operator<](#) (int \_\_lhs, [decimal32](#) \_\_rhs)
- [bool operator<](#) (int \_\_lhs, [decimal64](#) \_\_rhs)
- [bool operator<](#) (long \_\_lhs, [decimal128](#) \_\_rhs)
- [bool operator<](#) (long \_\_lhs, [decimal32](#) \_\_rhs)
- [bool operator<](#) (long \_\_lhs, [decimal64](#) \_\_rhs)
- [bool operator<](#) (long long \_\_lhs, [decimal128](#) \_\_rhs)
- [bool operator<](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [bool operator<](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [bool operator<](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [bool operator<](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [bool operator<](#) (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- [bool operator<](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)

- bool **operator**< (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal128 \_\_rhs)

- bool **operator**> (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned int \_\_rhs)

- `bool operator>= (decimal128 __lhs, unsigned long __rhs)`
- `bool operator>= (decimal128 __lhs, unsigned long long __rhs)`
- `bool operator>= (decimal32 __lhs, decimal128 __rhs)`
- `bool operator>= (decimal32 __lhs, decimal32 __rhs)`
- `bool operator>= (decimal32 __lhs, decimal64 __rhs)`
- `bool operator>= (decimal32 __lhs, int __rhs)`
- `bool operator>= (decimal32 __lhs, long __rhs)`
- `bool operator>= (decimal32 __lhs, long long __rhs)`
- `bool operator>= (decimal32 __lhs, unsigned int __rhs)`
- `bool operator>= (decimal32 __lhs, unsigned long __rhs)`
- `bool operator>= (decimal32 __lhs, unsigned long long __rhs)`
- `bool operator>= (decimal64 __lhs, decimal128 __rhs)`
- `bool operator>= (decimal64 __lhs, decimal32 __rhs)`
- `bool operator>= (decimal64 __lhs, decimal64 __rhs)`
- `bool operator>= (decimal64 __lhs, int __rhs)`
- `bool operator>= (decimal64 __lhs, long __rhs)`
- `bool operator>= (decimal64 __lhs, long long __rhs)`
- `bool operator>= (decimal64 __lhs, unsigned int __rhs)`
- `bool operator>= (decimal64 __lhs, unsigned long __rhs)`
- `bool operator>= (decimal64 __lhs, unsigned long long __rhs)`
- `bool operator>= (int __lhs, decimal128 __rhs)`
- `bool operator>= (int __lhs, decimal32 __rhs)`
- `bool operator>= (int __lhs, decimal64 __rhs)`
- `bool operator>= (long __lhs, decimal128 __rhs)`
- `bool operator>= (long __lhs, decimal32 __rhs)`
- `bool operator>= (long __lhs, decimal64 __rhs)`
- `bool operator>= (long long __lhs, decimal128 __rhs)`
- `bool operator>= (long long __lhs, decimal32 __rhs)`
- `bool operator>= (long long __lhs, decimal64 __rhs)`
- `bool operator>= (unsigned int __lhs, decimal128 __rhs)`
- `bool operator>= (unsigned int __lhs, decimal32 __rhs)`
- `bool operator>= (unsigned int __lhs, decimal64 __rhs)`
- `bool operator>= (unsigned long __lhs, decimal128 __rhs)`
- `bool operator>= (unsigned long __lhs, decimal32 __rhs)`
- `bool operator>= (unsigned long __lhs, decimal64 __rhs)`
- `bool operator>= (unsigned long long __lhs, decimal128 __rhs)`
- `bool operator>= (unsigned long long __lhs, decimal32 __rhs)`
- `bool operator>= (unsigned long long __lhs, decimal64 __rhs)`

#### 4.15.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

#### 4.15.2 Function Documentation

##### `decimal32_to_long_long()`

```
long long std::decimal::decimal32_to_long_long (
    decimal32 __d)
```

Non-conforming extension: Conversion to integral type.

## 4.16 std::experimental Namespace Reference

### Classes

- class [any](#)
- class [bad\\_any\\_cast](#)
- class [bad\\_optional\\_access](#)
- class [basic\\_string\\_view](#)
- struct [in\\_place\\_t](#)
- struct [nullopt\\_t](#)
- class [optional](#)
- class [ostream\\_joiner](#)
- struct [owner\\_less< shared\\_ptr< \\_Tp > >](#)
- struct [owner\\_less< weak\\_ptr< \\_Tp > >](#)
- class [propagate\\_const](#)

### Typedefs

- template<typename \_RAIter, typename \_Hash, typename \_Pred, typename \_Val = typename iterator\_traits<\_RAIter>::value\_type, typename \_Diff = typename iterator\_traits<\_RAIter>::difference\_type>  
using [\\_\\_boyer\\_moore\\_base\\_t](#)
- template<typename \_Tp>  
using [\\_\\_propagate\\_const\\_elem\\_type](#)
- using [erased\\_type](#)
- using [string\\_view](#)
- using [u16string\\_view](#)
- using [u32string\\_view](#)
- using [wstring\\_view](#)

### Functions

- template<typename \_Fn, typename \_Tuple, std::size\_t... \_Idx>  
constexpr decltype(auto) [\\_\\_apply\\_impl](#) (\_Fn &&\_\_f, \_Tuple &&\_\_t, [std::index\\_sequence< \\_Idx... >](#))
- template<typename \_Tp, size\_t \_Nm, size\_t... \_Idx>  
constexpr [array](#)< remove\_cv\_t< \_Tp >, \_Nm > [\\_\\_to\\_array](#) (\_Tp(&\_\_a)[\_Nm], [index\\_sequence< \\_Idx... >](#))
- [std::default\\_random\\_engine](#) & [S\\_randint\\_engine](#) ()
- template<typename \_ValueType>  
[\\_ValueType](#) [any\\_cast](#) (const [any](#) &\_\_any)
- template<typename \_Fn, typename \_Tuple>  
constexpr decltype(auto) [apply](#) (\_Fn &&\_\_f, \_Tuple &&\_\_t)
- template<typename \_Tp>  
bool [atomic\\_compare\\_exchange\\_strong](#) (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \*\_\_v, shared\_ptr< \_Tp > \_\_w)
- template<typename \_Tp>  
bool [atomic\\_compare\\_exchange\\_strong\\_explicit](#) (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \*\_\_v, shared\_ptr< \_Tp > \_\_w, [memory\\_order](#) \_\_success, [memory\\_order](#) \_\_failure)
- template<typename \_Tp>  
bool [atomic\\_compare\\_exchange\\_weak](#) (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \*\_\_v, shared\_ptr< \_Tp > \_\_w)
- template<typename \_Tp>  
bool [atomic\\_compare\\_exchange\\_weak\\_explicit](#) (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \*\_\_v, shared\_ptr< \_Tp > \_\_w, [memory\\_order](#) \_\_success, [memory\\_order](#) \_\_failure)
- template<typename \_Tp>  
void [atomic\\_exchange](#) (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \_\_r)

- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_exchange_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)`
- `template<typename _Tp>`  
`bool atomic_is_lock_free (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_load (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order __mo)`
- `template<typename _Tp>`  
`void atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_store_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Up>`  
`void erase (basic_string< _CharT, _Traits, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void erase (deque< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void erase (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void erase (list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void erase (vector< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Predicate>`  
`void erase_if (basic_string< _CharT, _Traits, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void erase_if (deque< _Tp, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void erase_if (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void erase_if (list< _Tp, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>`  
`void erase_if (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>`  
`void erase_if (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
`void erase_if (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
`void erase_if (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void erase_if (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void erase_if (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void erase_if (unordered_multiset< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void erase_if (unordered_set< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`



- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void erase_if (vector< _Tp, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Mn, typename _Nn>`  
`constexpr common_type_t< _Mn, _Nn > gcd (_Mn __m, _Nn __n) noexcept`
- `template<typename _Del, typename _Tp>`  
`_Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept`
- `template<typename _Tp>`  
`constexpr const _Tp & get_underlying (const propagate_const< _Tp > &__pt) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp & get_underlying (propagate_const< _Tp > &__pt) noexcept`
- `template<typename _Mn, typename _Nn>`  
`constexpr common_type_t< _Mn, _Nn > lcm (_Mn __m, _Nn __n)`
- `template<typename _Dest = void, typename... _Types>`  
`constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> make_array (↵  
_Types &&... __t)`
- `template<typename _RAlter, typename _Hash = std::hash<typename std::iterator_traits<_RAlter>::value_type>, typename _Binary↵  
Predicate = equal_to<>>>`  
`boyer_moore_horspool_searcher< _RAlter, _Hash, _BinaryPredicate > make_boyer_moore_horspool_searcher (  
_RAlter __pat_first, _RAlter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _RAlter, typename _Hash = std::hash<typename std::iterator_traits<_RAlter>::value_type>, typename _Binary↵  
Predicate = equal_to<>>>`  
`boyer_moore_searcher< _RAlter, _Hash, _BinaryPredicate > make_boyer_moore_searcher (_RAlter __pat ↵  
first, _RAlter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>>`  
`default_searcher< _ForwardIterator, _BinaryPredicate > make_default_searcher (_ForwardIterator __pat_first,   
_FowardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _Tp>`  
`constexpr observer_ptr< _Tp > make_observer (_Tp *__p) noexcept`
- `template<typename _CharT, typename _Traits, typename _DelimT>`  
`ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > make_ostream_joiner (basic_ostream< _CharT, ↵  
Traits > &__os, _DelimT &&__delimiter)`
- `template<typename _Fn>`  
`auto not_fn (_Fn &&__fn) noexcept(std::is_nothrow_constructible< std::decay_t< _Fn >, _Fn && >::value)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator!= (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view<   
_CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator!= (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view<   
_CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator!= (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits >   
__y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator!= (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator!= (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator!= (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr bool operator!= (const propagate_const< _Tp > &__pt, nullptr_t)`
- `template<typename _Tp>`  
`bool operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`



- `template<typename _Tp1, typename _Tp2>`  
`bool operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator!= (nullptr_t, const propagate_const< _Tp > &__pu)`
- `template<typename _Tp>`  
`bool operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator!= (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp>`  
`bool operator!= (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator!= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `constexpr basic_string_view< char > operator""sv (const char * __str, size_t __len) noexcept`
- `constexpr basic_string_view< char16_t > operator""sv (const char16_t * __str, size_t __len) noexcept`
- `constexpr basic_string_view< char32_t > operator""sv (const char32_t * __str, size_t __len) noexcept`
- `constexpr basic_string_view< wchar_t > operator""sv (const wchar_t * __str, size_t __len) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator< (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator< (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator< (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator< (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator< (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator< (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`bool operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator< (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, basic_string_view< _CharT, _Traits > __str)`
- `template<typename _Ch, typename _Tr, typename _Tp>`  
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator<= (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator<= (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`

- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator<= (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator<= (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator<= (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator<= (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`bool operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator<= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator== (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator== (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator== (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator== (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator== (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator== (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr bool operator== (const propagate_const< _Tp > &__pt, nullptr_t)`
- `template<typename _Tp>`  
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator== (nullptr_t, const propagate_const< _Tp > &__pu)`
- `template<typename _Tp>`  
`bool operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator== (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator== (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator== (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator> (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`

- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator> (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view<`  
`_CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator> (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits >`  
`__y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator> (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator> (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator> (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`bool operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator> (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator>= (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view<`  
`_CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator>= (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view<`  
`_CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator>= (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits >`  
`__y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator>= (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator>= (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator>= (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`bool operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator>= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _IntType>`  
`_IntType randint (_IntType __a, _IntType __b)`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `void reseed ()`
- `void reseed (default_random_engine::result_type __value)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance>`  
`_SampleIterator sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, ↵`  
`Distance __n)`

- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>  
_SampleIterator sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Searcher>  
_ForwardIterator search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`
- `template<typename _RandomAccessIterator>  
void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _Tp, typename _Tp1>  
shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `void swap (any &__x, any &__y) noexcept`
- `template<typename _Tp>  
constexpr void swap (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept`
- `template<typename _Tp>  
constexpr enable\_if\_t< __is_swappable< _Tp >::value, void > swap (propagate\_const< _Tp > &__pt, propagate\_const< _Tp > &__pt2) noexcept(!__is_nothrow_swappable< _Tp >::value)`
- `template<typename _Tp>  
void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp>  
void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, size_t _Nm>  
constexpr array< remove\_cv\_t< _Tp >, _Nm > to\_array (_Tp(&__a)[_Nm]) noexcept(is\_nothrow\_constructible< remove\_cv\_t< _Tp >, _Tp &::value)`
- `template<typename _ValueType>  
_ValueType any\_cast (any &__any)`
- `template<typename _ValueType, typename enable\_if<!__is_move_constructible< _ValueType >::value||is\_lvalue\_reference< _ValueType >::value, bool >::type = true>  
_ValueType any\_cast (any &&__any)`
- `template<typename _ValueType, typename enable\_if<__is_move_constructible< _ValueType >::value &&!__is_lvalue_reference< _ValueType >::value, bool >::type = false>  
_ValueType any\_cast (any &&__any)`
- `template<typename _ValueType>  
const _ValueType * any\_cast (const any *__any) noexcept`
- `template<typename _ValueType>  
_ValueType * any\_cast (any *__any) noexcept`

## Variables

- `template<typename _Yp, typename _Tp>  
constexpr bool __sp_compatible_v`
- `template<typename _Tp, typename _Yp>  
constexpr bool __sp_is_constructible_v`
- `constexpr in\_place\_t in\_place`
- `template<typename _Tp>  
constexpr bool is\_bind\_expression\_v`
- `template<typename _Tp>  
constexpr bool is_error_code_enum_v`
- `template<typename _Tp>  
constexpr bool is_error_condition_enum_v`

- `template<typename _Tp>`  
`constexpr int is\_placeholder\_v`
- `constexpr nullopt\_t nullopt`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_greater_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_greater_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_less_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_less_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_not_equal_v`
- `template<typename _Tp>`  
`constexpr size_t tuple_size_v`
  
- `template<typename _Tp>`  
`constexpr bool is_void_v`
- `template<typename _Tp>`  
`constexpr bool is_null_pointer_v`
- `template<typename _Tp>`  
`constexpr bool is_integral_v`
- `template<typename _Tp>`  
`constexpr bool is_floating_point_v`
- `template<typename _Tp>`  
`constexpr bool is_array_v`
- `template<typename _Tp>`  
`constexpr bool is_pointer_v`
- `template<typename _Tp>`  
`constexpr bool is_lvalue_reference_v`
- `template<typename _Tp>`  
`constexpr bool is_rvalue_reference_v`
- `template<typename _Tp>`  
`constexpr bool is_member_object_pointer_v`
- `template<typename _Tp>`  
`constexpr bool is_member_function_pointer_v`
- `template<typename _Tp>`  
`constexpr bool is_enum_v`
- `template<typename _Tp>`  
`constexpr bool is_union_v`
- `template<typename _Tp>`  
`constexpr bool is_class_v`
- `template<typename _Tp>`  
`constexpr bool is_function_v`
- `template<typename _Tp>`  
`constexpr bool is_reference_v`
- `template<typename _Tp>`  
`constexpr bool is_arithmetic_v`
- `template<typename _Tp>`  
`constexpr bool is_fundamental_v`

- `template<typename _Tp>`  
`constexpr bool is_object_v`
- `template<typename _Tp>`  
`constexpr bool is_scalar_v`
- `template<typename _Tp>`  
`constexpr bool is_compound_v`
- `template<typename _Tp>`  
`constexpr bool is_member_pointer_v`
- `template<typename _Tp>`  
`constexpr bool is_const_v`
- `template<typename _Tp>`  
`constexpr bool is_volatile_v`
- `template<typename _Tp>`  
`constexpr bool is_trivial_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_copyable_v`
- `template<typename _Tp>`  
`constexpr bool is_standard_layout_v`
- `template<typename _Tp>`  
`constexpr bool is_pod_v`
- `template<typename _Tp>`  
`constexpr bool is_literal_type_v`
- `template<typename _Tp>`  
`constexpr bool is_empty_v`
- `template<typename _Tp>`  
`constexpr bool is_polymorphic_v`
- `template<typename _Tp>`  
`constexpr bool is_abstract_v`
- `template<typename _Tp>`  
`constexpr bool is_final_v`
- `template<typename _Tp>`  
`constexpr bool is_signed_v`
- `template<typename _Tp>`  
`constexpr bool is_unsigned_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool is_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool is_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_destructible_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool is_trivially_constructible_v`

- `template<typename _Tp>`  
`constexpr bool is_trivially_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool is_trivially_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_destructible_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool is_nothrow_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool is_nothrow_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_destructible_v`
- `template<typename _Tp>`  
`constexpr bool has_virtual_destructor_v`
- `template<typename _Tp>`  
`constexpr size_t alignment_of_v`
- `template<typename _Tp>`  
`constexpr size_t rank_v`
- `template<typename _Tp, unsigned _Idx = 0>`  
`constexpr size_t extent_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool is_same_v`
- `template<typename _Tp>`  
`constexpr bool is_same_v< _Tp, _Tp >`
- `template<typename _Base, typename _Derived>`  
`constexpr bool is_base_of_v`
- `template<typename _From, typename _To>`  
`constexpr bool is_convertible_v`
  
- `template<typename... _Bn>`  
`constexpr bool conjunction_v`
- `template<typename... _Bn>`  
`constexpr bool disjunction_v`

- `template<typename _Pp>`  
`constexpr bool negation_v`
- `template<typename...>`  
`using void_t`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using detected_or`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using detected_or_t`
- `template<template< typename... > class _Op, typename... _Args>`  
`using detected_t`
- `template<template< typename... > class _Op, typename... _Args>`  
`using is_detected`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`using is_detected_exact`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`using is_detected_convertible`
- `template<template< typename... > class _Op, typename... _Args>`  
`constexpr bool is_detected_v`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`constexpr bool is_detected_exact_v`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`constexpr bool is_detected_convertible_v`

#### 4.16.1 Detailed Description

Namespace for features defined in ISO Technical Specifications.

#### 4.16.2 Function Documentation

##### **gcd()**

```
template<typename _Mn, typename _Nn>
common_type_t< _Mn, _Nn > std::experimental::fundamentals_v2::gcd (
    _Mn __m,
    _Nn __n) [constexpr], [noexcept]
```

Greatest common divisor.

##### **get\_deleter()**

```
template<typename _Del, typename _Tp>
_Del * std::experimental::fundamentals_v2::get_deleter (
    const shared_ptr< _Tp > & __p) [inline], [noexcept]
```

C++14 20.8.2.2.10.

##### **lcm()**

```
template<typename _Mn, typename _Nn>
common_type_t< _Mn, _Nn > std::experimental::fundamentals_v2::lcm (
    _Mn __m,
    _Nn __n) [constexpr]
```

Least common multiple.



**make\_boyer\_moore\_horspool\_searcher()**

```
template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>,
typename _BinaryPredicate = equal_to<>>>
boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::fundamentals_v1::make_boyer_moore_horspool_searcher (
    _RAIter __pat_first,
    _RAIter __pat_last,
    _Hash __hf = _Hash(),
    _BinaryPredicate __pred = _BinaryPredicate()) [inline]
```

Generator function for boyer\_moore\_horspool\_searcher.

**make\_boyer\_moore\_searcher()**

```
template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>,
typename _BinaryPredicate = equal_to<>>>
boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::fundamentals_v1::make_boyer_moore_searcher (
    _RAIter __pat_first,
    _RAIter __pat_last,
    _Hash __hf = _Hash(),
    _BinaryPredicate __pred = _BinaryPredicate()) [inline]
```

Generator function for boyer\_moore\_searcher.

**make\_default\_searcher()**

```
template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>
default_searcher< _ForwardIterator, _BinaryPredicate > std::experimental::fundamentals_v1::make_default_searcher (
    _ForwardIterator __pat_first,
    _ForwardIterator __pat_last,
    _BinaryPredicate __pred = _BinaryPredicate()) [inline]
```

Generator function for default\_searcher.

**make\_ostream\_joiner()**

```
template<typename _CharT, typename _Traits, typename _DelimT>
ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > std::experimental::fundamentals_v2::make_ostream_joiner (
    basic_ostream< _CharT, _Traits > & __os,
    _DelimT && __delimiter) [inline]
```

Object generator for ostream\_joiner.

**not\_fn()**

```
template<typename _Fn>
auto std::experimental::fundamentals_v2::not_fn (
    _Fn && __fn) [inline], [noexcept]
```

[func.not\_fn] Function template not\_fn

**sample()**

```
template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>
_SampleIterator std::experimental::fundamentals_v2::sample (
```

```

_PopulationIterator __first,
_PopulationIterator __last,
_SampleIterator __out,
_Distance __n,
_UniformRandomNumberGenerator && __g)

```

Take a random sample from a population.

#### 4.16.3 Variable Documentation

##### `is_bind_expression_v`

```

template<typename _Tp>
bool std::experimental::fundamentals_v1::is_bind_expression_v [constexpr]

```

Variable template for `std::is_bind_expression`.

##### `is_placeholder_v`

```

template<typename _Tp>
int std::experimental::fundamentals_v1::is_placeholder_v [constexpr]

```

Variable template for `std::is_placeholder`.

## 4.17 `std::filesystem` Namespace Reference

### Classes

- class [directory\\_entry](#)
- class [directory\\_iterator](#)
- class [file\\_status](#)
- class [filesystem\\_error](#)
- class [path](#)
- class [recursive\\_directory\\_iterator](#)
- struct [space\\_info](#)

### Typedefs

- using [file\\_time\\_type](#)

### Enumerations

- enum class [copy\\_options](#) : unsigned short { **none**, **skip\_existing**, **overwrite\_existing**, **update\_existing**, **recursive**, **copy\_symlinks**, **skip\_symlinks**, **directories\_only**, **create\_symlinks**, **create\_hard\_links** }
- enum class [directory\\_options](#) : unsigned char { **none**, **follow\_directory\_symlink**, **skip\_permission\_denied** }
- enum class [file\\_type](#) : signed char { **none**, **not\_found**, **regular**, **directory**, **symlink**, **block**, **character**, **fifo**, **socket**, **unknown** }
- enum class [perm\\_options](#) : unsigned { **replace**, **add**, **remove**, **nofollow** }
- enum class [perms](#) : unsigned { **none**, **owner\_read**, **owner\_write**, **owner\_exec**, **owner\_all**, **group\_read**, **group\_write**, **group\_exec**, **group\_all**, **others\_read**, **others\_write**, **others\_exec**, **others\_all**, **all**, **set\_uid**, **set\_gid**, **sticky\_bit**, **mask**, **unknown** }

## Functions

- **path absolute** (const **path** &\_\_p)
- **path absolute** (const **path** &\_\_p, **error\_code** &\_\_ec)
- **path canonical** (const **path** &\_\_p)
- **path canonical** (const **path** &\_\_p, **error\_code** &\_\_ec)
- void **copy** (const **path** &\_\_from, const **path** &\_\_to)
- void **copy** (const **path** &\_\_from, const **path** &\_\_to, **copy\_options** \_\_options)
- void **copy** (const **path** &\_\_from, const **path** &\_\_to, **copy\_options** \_\_options, **error\_code** &\_\_ec)
- void **copy** (const **path** &\_\_from, const **path** &\_\_to, **error\_code** &\_\_ec)
- bool **copy\_file** (const **path** &\_\_from, const **path** &\_\_to)
- bool **copy\_file** (const **path** &\_\_from, const **path** &\_\_to, **copy\_options** \_\_option)
- bool **copy\_file** (const **path** &\_\_from, const **path** &\_\_to, **copy\_options** \_\_option, **error\_code** &\_\_ec)
- bool **copy\_file** (const **path** &\_\_from, const **path** &\_\_to, **error\_code** &\_\_ec)
- void **copy\_symlink** (const **path** &\_\_existing\_symlink, const **path** &\_\_new\_symlink)
- void **copy\_symlink** (const **path** &\_\_existing\_symlink, const **path** &\_\_new\_symlink, **error\_code** &\_\_ec) noexcept
- bool **create\_directories** (const **path** &\_\_p)
- bool **create\_directories** (const **path** &\_\_p, **error\_code** &\_\_ec)
- bool **create\_directory** (const **path** &\_\_p)
- bool **create\_directory** (const **path** &\_\_p, const **path** &\_\_attributes)
- bool **create\_directory** (const **path** &\_\_p, const **path** &\_\_attributes, **error\_code** &\_\_ec) noexcept
- bool **create\_directory** (const **path** &\_\_p, **error\_code** &\_\_ec) noexcept
- void **create\_directory\_symlink** (const **path** &\_\_to, const **path** &\_\_new\_symlink)
- void **create\_directory\_symlink** (const **path** &\_\_to, const **path** &\_\_new\_symlink, **error\_code** &\_\_ec) noexcept
- void **create\_hard\_link** (const **path** &\_\_to, const **path** &\_\_new\_hard\_link)
- void **create\_hard\_link** (const **path** &\_\_to, const **path** &\_\_new\_hard\_link, **error\_code** &\_\_ec) noexcept
- void **create\_symlink** (const **path** &\_\_to, const **path** &\_\_new\_symlink)
- void **create\_symlink** (const **path** &\_\_to, const **path** &\_\_new\_symlink, **error\_code** &\_\_ec) noexcept
- **path current\_path** ()
- void **current\_path** (const **path** &\_\_p)
- void **current\_path** (const **path** &\_\_p, **error\_code** &\_\_ec) noexcept
- **path current\_path** (**error\_code** &\_\_ec)
- bool **equivalent** (const **path** &\_\_p1, const **path** &\_\_p2)
- bool **equivalent** (const **path** &\_\_p1, const **path** &\_\_p2, **error\_code** &\_\_ec) noexcept
- bool **exists** (const **path** &\_\_p)
- bool **exists** (const **path** &\_\_p, **error\_code** &\_\_ec) noexcept
- bool **exists** (**file\_status**) noexcept
- **uintmax\_t file\_size** (const **path** &\_\_p)
- **uintmax\_t file\_size** (const **path** &\_\_p, **error\_code** &\_\_ec) noexcept
- **uintmax\_t hard\_link\_count** (const **path** &\_\_p)
- **uintmax\_t hard\_link\_count** (const **path** &\_\_p, **error\_code** &\_\_ec) noexcept
- **size\_t hash\_value** (const **path** &\_\_p) noexcept
- bool **is\_block\_file** (const **path** &\_\_p)
- bool **is\_block\_file** (const **path** &\_\_p, **error\_code** &\_\_ec) noexcept
- bool **is\_block\_file** (**file\_status** \_\_s) noexcept
- bool **is\_character\_file** (const **path** &\_\_p)
- bool **is\_character\_file** (const **path** &\_\_p, **error\_code** &\_\_ec) noexcept
- bool **is\_character\_file** (**file\_status** \_\_s) noexcept
- bool **is\_directory** (const **path** &\_\_p)
- bool **is\_directory** (const **path** &\_\_p, **error\_code** &\_\_ec) noexcept
- bool **is\_directory** (**file\_status** \_\_s) noexcept

- `bool is_empty (const path &__p)`
- `bool is_empty (const path &__p, error_code &__ec)`
- `bool is_fifo (const path &__p)`
- `bool is_fifo (const path &__p, error_code &__ec) noexcept`
- `bool is_fifo (file_status __s) noexcept`
- `bool is_other (const path &__p)`
- `bool is_other (const path &__p, error_code &__ec) noexcept`
- `bool is_other (file_status) noexcept`
- `bool is_regular_file (const path &__p)`
- `bool is_regular_file (const path &__p, error_code &__ec) noexcept`
- `bool is_regular_file (file_status) noexcept`
- `bool is_socket (const path &__p)`
- `bool is_socket (const path &__p, error_code &__ec) noexcept`
- `bool is_socket (file_status __s) noexcept`
- `bool is_symlink (const path &__p)`
- `bool is_symlink (const path &__p, error_code &__ec) noexcept`
- `bool is_symlink (file_status) noexcept`
- `file_time_type last_write_time (const path &)`
- `file_time_type last_write_time (const path &, error_code &) noexcept`
- `void last_write_time (const path &__p, file_time_type __new_time)`
- `void last_write_time (const path &__p, file_time_type __new_time, error_code &__ec) noexcept`
- `copy_options & operator&= (copy_options &__x, copy_options __y) noexcept`
- `constexpr copy_options operator^ (copy_options __x, copy_options __y) noexcept`
- `copy_options & operator^= (copy_options &__x, copy_options __y) noexcept`
- `constexpr copy_options operator| (copy_options __x, copy_options __y) noexcept`
- `copy_options & operator|= (copy_options &__x, copy_options __y) noexcept`
- `constexpr copy_options operator~ (copy_options __x) noexcept`
- `void permissions (const path &, perms, perm_options, error_code &) noexcept`
- `void permissions (const path &__p, perms __prms, error_code &__ec) noexcept`
- `void permissions (const path &__p, perms __prms, perm_options __opts=perm_options::replace)`
- `path proximate (const path &__p, const path &__base, error_code &__ec)`
- `path proximate (const path &__p, const path &__base=current_path())`
- `path proximate (const path &__p, error_code &__ec)`
- `path read_symlink (const path &__p)`
- `path read_symlink (const path &__p, error_code &__ec)`
- `path relative (const path &__p, const path &__base, error_code &__ec)`
- `path relative (const path &__p, const path &__base=current_path())`
- `path relative (const path &__p, error_code &__ec)`
- `bool remove (const path &, error_code &) noexcept`
- `bool remove (const path &__p)`
- `uintmax_t remove_all (const path &)`
- `uintmax_t remove_all (const path &, error_code &)`
- `void rename (const path &__from, const path &__to)`
- `void rename (const path &__from, const path &__to, error_code &__ec) noexcept`
- `void resize_file (const path &__p, uintmax_t __size)`
- `void resize_file (const path &__p, uintmax_t __size, error_code &__ec) noexcept`
- `space_info space (const path &__p)`
- `space_info space (const path &__p, error_code &__ec) noexcept`
- `file_status status (const path &)`
- `file_status status (const path &, error_code &) noexcept`
- `bool status_known (file_status) noexcept`

- void **swap** ([path](#) &\_\_lhs, [path](#) &\_\_rhs) noexcept
  - [file\\_status](#) **symlink\_status** (const [path](#) &)
  - [file\\_status](#) **symlink\_status** (const [path](#) &, [error\\_code](#) &) noexcept
  - [path](#) **temp\_directory\_path** ()
  - [path](#) **temp\_directory\_path** ([error\\_code](#) &\_\_ec)
  - template<typename \_InputIterator, typename \_Require = \_\_detail::\_\_Path2<\_InputIterator>, typename \_CharT = \_\_detail::\_\_value\_type\_<↔ is\_char\_or\_char8\_t<\_InputIterator>>>  
[path](#) **u8path** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - template<typename \_Source, typename \_Require = \_\_detail::\_\_Path<\_Source>, typename \_CharT = \_\_detail::\_\_value\_type\_is\_char\_or\_↔\_char8\_t<\_Source>>>  
[path](#) **u8path** (const \_Source &\_\_source)
  - [path](#) **weakly\_canonical** (const [path](#) &\_\_p)
  - [path](#) **weakly\_canonical** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- 
- constexpr [perms](#) **operator|** ([perms](#) \_\_x, [perms](#) \_\_y) noexcept
  - constexpr [perms](#) **operator^** ([perms](#) \_\_x, [perms](#) \_\_y) noexcept
  - constexpr [perms](#) **operator~** ([perms](#) \_\_x) noexcept
  - [perms](#) & **operator&=** ([perms](#) &\_\_x, [perms](#) \_\_y) noexcept
  - [perms](#) & **operator|=** ([perms](#) &\_\_x, [perms](#) \_\_y) noexcept
  - [perms](#) & **operator^=** ([perms](#) &\_\_x, [perms](#) \_\_y) noexcept
- 
- constexpr [perm\\_options](#) **operator|** ([perm\\_options](#) \_\_x, [perm\\_options](#) \_\_y) noexcept
  - constexpr [perm\\_options](#) **operator^** ([perm\\_options](#) \_\_x, [perm\\_options](#) \_\_y) noexcept
  - constexpr [perm\\_options](#) **operator~** ([perm\\_options](#) \_\_x) noexcept
  - [perm\\_options](#) & **operator&=** ([perm\\_options](#) &\_\_x, [perm\\_options](#) \_\_y) noexcept
  - [perm\\_options](#) & **operator|=** ([perm\\_options](#) &\_\_x, [perm\\_options](#) \_\_y) noexcept
  - [perm\\_options](#) & **operator^=** ([perm\\_options](#) &\_\_x, [perm\\_options](#) \_\_y) noexcept
- 
- constexpr [directory\\_options](#) **operator|** ([directory\\_options](#) \_\_x, [directory\\_options](#) \_\_y) noexcept
  - constexpr [directory\\_options](#) **operator^** ([directory\\_options](#) \_\_x, [directory\\_options](#) \_\_y) noexcept
  - constexpr [directory\\_options](#) **operator~** ([directory\\_options](#) \_\_x) noexcept
  - [directory\\_options](#) & **operator&=** ([directory\\_options](#) &\_\_x, [directory\\_options](#) \_\_y) noexcept
  - [directory\\_options](#) & **operator|=** ([directory\\_options](#) &\_\_x, [directory\\_options](#) \_\_y) noexcept
  - [directory\\_options](#) & **operator^=** ([directory\\_options](#) &\_\_x, [directory\\_options](#) \_\_y) noexcept

#### 4.17.1 Detailed Description

ISO C++ 2017 namespace for File System library.

### 4.18 std::literals Namespace Reference

#### Namespaces

- namespace [chrono\\_literals](#)

**Functions**

- constexpr `chrono::day` `operator""d` (unsigned long long \_\_d) noexcept
- template<char... \_Digits>  
constexpr `chrono::hours` `operator""h` ()
- constexpr `chrono::duration`< long double, `ratio`< 3600, 1 > > `operator""h` (long double \_\_hours)
- template<char... \_Digits>  
constexpr `chrono::minutes` `operator""min` ()
- constexpr `chrono::duration`< long double, `ratio`< 60, 1 > > `operator""min` (long double \_\_mins)
- template<char... \_Digits>  
constexpr `chrono::milliseconds` `operator""ms` ()
- constexpr `chrono::duration`< long double, `milli` > `operator""ms` (long double \_\_msecs)
- template<char... \_Digits>  
constexpr `chrono::nanoseconds` `operator""ns` ()
- constexpr `chrono::duration`< long double, `nano` > `operator""ns` (long double \_\_nsecs)
- template<char... \_Digits>  
constexpr `chrono::seconds` `operator""s` ()
- constexpr `chrono::duration`< long double > `operator""s` (long double \_\_secs)
- constexpr `basic_string_view`< char > `operator""sv` (const char \* \_\_str, size\_t \_\_len) noexcept
- constexpr `basic_string_view`< char16\_t > `operator""sv` (const char16\_t \* \_\_str, size\_t \_\_len) noexcept
- constexpr `basic_string_view`< char32\_t > `operator""sv` (const char32\_t \* \_\_str, size\_t \_\_len) noexcept
- constexpr `basic_string_view`< wchar\_t > `operator""sv` (const wchar\_t \* \_\_str, size\_t \_\_len) noexcept
- template<char... \_Digits>  
constexpr `chrono::microseconds` `operator""us` ()
- constexpr `chrono::duration`< long double, `micro` > `operator""us` (long double \_\_usecs)
- constexpr `chrono::year` `operator""y` (unsigned long long \_\_y) noexcept

**4.18.1 Detailed Description**

ISO C++ inline namespace for literal suffixes.

**4.19 `std::literals::chrono_literals` Namespace Reference****Functions**

- constexpr `chrono::day` `operator""d` (unsigned long long \_\_d) noexcept
- template<char... \_Digits>  
constexpr `chrono::hours` `operator""h` ()
- constexpr `chrono::duration`< long double, `ratio`< 3600, 1 > > `operator""h` (long double \_\_hours)
- template<char... \_Digits>  
constexpr `chrono::minutes` `operator""min` ()
- constexpr `chrono::duration`< long double, `ratio`< 60, 1 > > `operator""min` (long double \_\_mins)
- template<char... \_Digits>  
constexpr `chrono::milliseconds` `operator""ms` ()
- constexpr `chrono::duration`< long double, `milli` > `operator""ms` (long double \_\_msecs)
- template<char... \_Digits>  
constexpr `chrono::nanoseconds` `operator""ns` ()
- constexpr `chrono::duration`< long double, `nano` > `operator""ns` (long double \_\_nsecs)
- template<char... \_Digits>  
constexpr `chrono::seconds` `operator""s` ()
- constexpr `chrono::duration`< long double > `operator""s` (long double \_\_secs)
- template<char... \_Digits>  
constexpr `chrono::microseconds` `operator""us` ()
- constexpr `chrono::duration`< long double, `micro` > `operator""us` (long double \_\_usecs)
- constexpr `chrono::year` `operator""y` (unsigned long long \_\_y) noexcept

#### 4.19.1 Detailed Description

ISO C++ 2014 namespace for suffixes for duration literals.

These suffixes can be used to create `chrono::duration` values with tick periods of hours, minutes, seconds, milliseconds, microseconds or nanoseconds. For example, `std::chrono::seconds(5)` can be written as `5s` after making the suffix visible in the current scope. The suffixes can be made visible by a `using-directive` or `using-declaration` such as:

```
• using namespace std::chrono_literals;
• using namespace std::literals;
• using namespace std::chrono;
• using namespace std;
• using std::chrono_literals::operator"s;
```

The result of these suffixes on an integer literal is one of the standard typedefs such as `std::chrono::hours`. The result on a floating-point literal is a duration type with the specified tick period and an unspecified floating-point representation, for example `1.5e2ms` might be equivalent to `chrono::duration<long double, chrono::milli>(1.5e2)`.

Since

C+14

## 4.20 std::placeholders Namespace Reference

### Variables

- const [\\_Placeholder](#)< 1 > [\\_1](#)
- const [\\_Placeholder](#)< 10 > [\\_10](#)
- const [\\_Placeholder](#)< 11 > [\\_11](#)
- const [\\_Placeholder](#)< 12 > [\\_12](#)
- const [\\_Placeholder](#)< 13 > [\\_13](#)
- const [\\_Placeholder](#)< 14 > [\\_14](#)
- const [\\_Placeholder](#)< 15 > [\\_15](#)
- const [\\_Placeholder](#)< 16 > [\\_16](#)
- const [\\_Placeholder](#)< 17 > [\\_17](#)
- const [\\_Placeholder](#)< 18 > [\\_18](#)
- const [\\_Placeholder](#)< 19 > [\\_19](#)
- const [\\_Placeholder](#)< 2 > [\\_2](#)
- const [\\_Placeholder](#)< 20 > [\\_20](#)
- const [\\_Placeholder](#)< 21 > [\\_21](#)
- const [\\_Placeholder](#)< 22 > [\\_22](#)
- const [\\_Placeholder](#)< 23 > [\\_23](#)
- const [\\_Placeholder](#)< 24 > [\\_24](#)
- const [\\_Placeholder](#)< 25 > [\\_25](#)
- const [\\_Placeholder](#)< 26 > [\\_26](#)
- const [\\_Placeholder](#)< 27 > [\\_27](#)
- const [\\_Placeholder](#)< 28 > [\\_28](#)
- const [\\_Placeholder](#)< 29 > [\\_29](#)
- const [\\_Placeholder](#)< 3 > [\\_3](#)
- const [\\_Placeholder](#)< 4 > [\\_4](#)
- const [\\_Placeholder](#)< 5 > [\\_5](#)
- const [\\_Placeholder](#)< 6 > [\\_6](#)
- const [\\_Placeholder](#)< 7 > [\\_7](#)
- const [\\_Placeholder](#)< 8 > [\\_8](#)
- const [\\_Placeholder](#)< 9 > [\\_9](#)

### 4.20.1 Detailed Description

ISO C++ 2011 namespace for `std::bind` placeholders.

Since

C++11

## 4.21 `std::regex_constants` Namespace Reference

### 5.1 Regular Expression Syntax Options

- enum `syntax_option_type` : unsigned int {  
`_S_icalse` , `_S_nosubs` , `_S_optimize` , `_S_collate` ,  
`_S_ECMAScript` , `_S_basic` , `_S_extended` , `_S_awk` ,  
`_S_grep` , `_S_egrep` , `_S_polynomial` , `_S_multiline` }
- constexpr `syntax_option_type` `icalse`
- constexpr `syntax_option_type` `nosubs`
- constexpr `syntax_option_type` `optimize`
- constexpr `syntax_option_type` `collate`
- constexpr `syntax_option_type` `ECMAScript`
- constexpr `syntax_option_type` `basic`
- constexpr `syntax_option_type` `extended`
- constexpr `syntax_option_type` `awk`
- constexpr `syntax_option_type` `grep`
- constexpr `syntax_option_type` `egrep`
- constexpr `syntax_option_type` `multiline`
- constexpr `syntax_option_type` `__multiline`
- constexpr `syntax_option_type` `__polynomial`
- constexpr `syntax_option_type` `operator&` (`syntax_option_type` \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` `operator|` (`syntax_option_type` \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` `operator^` (`syntax_option_type` \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` `operator~` (`syntax_option_type` \_\_a) noexcept
- constexpr `syntax_option_type` & `operator&=` (`syntax_option_type` & \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` & `operator|=` (`syntax_option_type` & \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` & `operator^=` (`syntax_option_type` & \_\_a, `syntax_option_type` \_\_b) noexcept

### 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `match_flag_type` : unsigned int {  
`_S_default` , `_S_not_bol` , `_S_not_eol` , `_S_not_bow` ,  
`_S_not_eow` , `_S_any` , `_S_not_null` , `_S_continuous` ,  
`_S_prev_avail` , `_S_sed` , `_S_no_copy` , `_S_first_only` ,  
`_S_match_flag_last` }
- constexpr `match_flag_type` `match_default`
- constexpr `match_flag_type` `match_not_bol`
- constexpr `match_flag_type` `match_not_eol`
- constexpr `match_flag_type` `match_not_bow`
- constexpr `match_flag_type` `match_not_eow`
- constexpr `match_flag_type` `match_any`



- constexpr `match_flag_type` `match_not_null`
- constexpr `match_flag_type` `match_continuous`
- constexpr `match_flag_type` `match_prev_avail`
- constexpr `match_flag_type` `format_default`
- constexpr `match_flag_type` `format_sed`
- constexpr `match_flag_type` `format_no_copy`
- constexpr `match_flag_type` `format_first_only`
- constexpr `match_flag_type` `operator&` (`match_flag_type` \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` `operator|` (`match_flag_type` \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` `operator^` (`match_flag_type` \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` `operator~` (`match_flag_type` \_\_a) noexcept
- constexpr `match_flag_type` & `operator&=` (`match_flag_type` &\_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` & `operator|=` (`match_flag_type` &\_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` & `operator^=` (`match_flag_type` &\_\_a, `match_flag_type` \_\_b) noexcept

### 5.3 Error Types

- enum `error_type` {  
`_S_error_collate`, `_S_error_ctype`, `_S_error_escape`, `_S_error_backref`,  
`_S_error_brack`, `_S_error_paren`, `_S_error_brace`, `_S_error_badbrace`,  
`_S_error_range`, `_S_error_space`, `_S_error_badrepeat`, `_S_error_complexity`,  
`_S_error_stack`, `_S_null`, `_S_grammar` }
- constexpr `error_type` `error_collate` (`_S_error_collate`)
- constexpr `error_type` `error_ctype` (`_S_error_ctype`)
- constexpr `error_type` `error_escape` (`_S_error_escape`)
- constexpr `error_type` `error_backref` (`_S_error_backref`)
- constexpr `error_type` `error_brack` (`_S_error_brack`)
- constexpr `error_type` `error_paren` (`_S_error_paren`)
- constexpr `error_type` `error_brace` (`_S_error_brace`)
- constexpr `error_type` `error_badbrace` (`_S_error_badbrace`)
- constexpr `error_type` `error_range` (`_S_error_range`)
- constexpr `error_type` `error_space` (`_S_error_space`)
- constexpr `error_type` `error_badrepeat` (`_S_error_badrepeat`)
- constexpr `error_type` `error_complexity` (`_S_error_complexity`)
- constexpr `error_type` `error_stack` (`_S_error_stack`)

#### 4.21.1 Detailed Description

ISO C++ 2011 namespace for options and flags used with `std::regex`.

#### 4.21.2 Enumeration Type Documentation

##### `error_type`

```
enum std::regex_constants::error_type
```

The expression contained an invalid collating element name.

##### `match_flag_type`

```
enum std::regex_constants::match_flag_type : unsigned int
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

### syntax\_option\_type

```
enum std::regex_constants::syntax_option_type : unsigned int
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

#### 4.21.3 Function Documentation

##### error\_backref()

```
error_type std::regex_constants::error_backref (
    _S_error_backref ) [inline], [constexpr]
```

The expression contained an invalid back reference.

##### error\_badbrace()

```
error_type std::regex_constants::error_badbrace (
    _S_error_badbrace ) [inline], [constexpr]
```

The expression contained an invalid range in a `{}` expression.

##### error\_badrepeat()

```
error_type std::regex_constants::error_badrepeat (
    _S_error_badrepeat ) [inline], [constexpr]
```

One of `*?+{` was not preceded by a valid regular expression.

##### error\_brace()

```
error_type std::regex_constants::error_brace (
    _S_error_brace ) [inline], [constexpr]
```

The expression contained mismatched `{` and `}`

##### error\_brack()

```
error_type std::regex_constants::error_brack (
    _S_error_brack ) [inline], [constexpr]
```

The expression contained mismatched `[` and `]`.

##### error\_collate()

```
error_type std::regex_constants::error_collate (
    _S_error_collate ) [inline], [constexpr]
```

The expression contained an invalid collating element name.

##### error\_complexity()

```
error_type std::regex_constants::error_complexity (
    _S_error_complexity ) [inline], [constexpr]
```

The complexity of an attempted match against a regular expression exceeded a pre-set level.

**error\_ctype()**

```
error_type std::regex_constants::error_ctype (
    _S_error_ctype ) [inline], [constexpr]
```

The expression contained an invalid character class name.

**error\_escape()**

```
error_type std::regex_constants::error_escape (
    _S_error_escape ) [inline], [constexpr]
```

The expression contained an invalid escaped character, or a trailing escape.

**error\_paren()**

```
error_type std::regex_constants::error_paren (
    _S_error_paren ) [inline], [constexpr]
```

The expression contained mismatched ( and ).

**error\_range()**

```
error_type std::regex_constants::error_range (
    _S_error_range ) [inline], [constexpr]
```

The expression contained an invalid character range, such as [b-a] in most encodings.

**error\_space()**

```
error_type std::regex_constants::error_space (
    _S_error_space ) [inline], [constexpr]
```

There was insufficient memory to convert the expression into a finite state machine.

**error\_stack()**

```
error_type std::regex_constants::error_stack (
    _S_error_stack ) [inline], [constexpr]
```

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

**operator&() [1/2]**

```
match_flag_type std::regex_constants::operator& (
    match_flag_type __a,
    match_flag_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

**operator&() [2/2]**

```
syntax_option_type std::regex_constants::operator& (
    syntax_option_type __a,
    syntax_option_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

**operator&=()** [1/2]

```
match_flag_type & std::regex_constants::operator&= (
    match_flag_type & __a,
    match_flag_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

**operator&=()** [2/2]

```
syntax_option_type & std::regex_constants::operator&= (
    syntax_option_type & __a,
    syntax_option_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

**operator^()** [1/2]

```
match_flag_type std::regex_constants::operator^ (
    match_flag_type __a,
    match_flag_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

**operator^()** [2/2]

```
syntax_option_type std::regex_constants::operator^ (
    syntax_option_type __a,
    syntax_option_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

**operator^=()** [1/2]

```
match_flag_type & std::regex_constants::operator^= (
    match_flag_type & __a,
    match_flag_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

**operator^=()** [2/2]

```
syntax_option_type & std::regex_constants::operator^= (
    syntax_option_type & __a,
    syntax_option_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

#### **operator" |()** [1/2]

```
match_flag_type std::regex_constants::operator| (
    match_flag_type __a,
    match_flag_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

#### **operator" |()** [2/2]

```
syntax_option_type std::regex_constants::operator| (
    syntax_option_type __a,
    syntax_option_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

#### **operator" |=()** [1/2]

```
match_flag_type & std::regex_constants::operator|= (
    match_flag_type & __a,
    match_flag_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

#### **operator" |=()** [2/2]

```
syntax_option_type & std::regex_constants::operator|= (
    syntax_option_type & __a,
    syntax_option_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

#### **operator~()** [1/2]

```
match_flag_type std::regex_constants::operator~ (
    match_flag_type __a) [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

**operator~()** [2/2]

```
syntax_option_type std::regex_constants::operator~ (
    syntax_option_type __a) [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

**4.21.4 Variable Documentation****\_\_multiline**

```
syntax_option_type std::regex_constants::__multiline [inline], [constexpr]
```

Extension: Equivalent to `regex_constants::multiline` for C++11 and C++14.

**\_\_polynomial**

```
syntax_option_type std::regex_constants::__polynomial [inline], [constexpr]
```

Extension: Ensure both space complexity of compiled regex and time complexity execution are not exponential. If specified in a regex with back-references, the exception `regex_constants::error_complexity` will be thrown.

**awk**

```
syntax_option_type std::regex_constants::awk [inline], [constexpr]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `awk` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type extended`, except that C-style escape sequences are supported. These sequences are: `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\&apos;`, `\&apos;`, and `\ddd` (where `ddd` is one, two, or three octal digits).

**basic**

```
syntax_option_type std::regex_constants::basic [inline], [constexpr]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

**collate**

```
syntax_option_type std::regex_constants::collate [inline], [constexpr]
```

Specifies that character ranges of the form `[a-b]` should be locale sensitive.

**ECMAScript**

```
syntax_option_type std::regex_constants::ECMAScript [inline], [constexpr]
```

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

**egrep**

```
syntax_option_type std::regex_constants::egrep [inline], [constexpr]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` when given the `-E` option in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that newlines are treated as whitespace.

### extended

`syntax_option_type` `std::regex_constants::extended` `[inline]`, `[constexpr]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

### format\_default

`match_flag_type` `std::regex_constants::format_default` `[inline]`, `[constexpr]`

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript `replace` function in ECMA- 262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 `String.prototype.replace`. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- `$$` The dollar-sign itself (`$`)
- `$&` The matched substring.
- `$'` The portion of *string* that precedes the matched substring. This would be `match_results::prefix()`.
- `$'` The portion of *string* that follows the matched substring. This would be `match_results::suffix()`.
- `$n` The *n*th capture, where *n* is in `[1,9]` and `$n` is not followed by a decimal digit. If `n <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.
- `$nn` The *nn*th capture, where *nn* is a two-digit decimal number on `[01, 99]`. If `nn <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `nn > match_results::size()`, the result is implementation-defined.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, and `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`.

### format\_first\_only

`match_flag_type` `std::regex_constants::format_first_only` `[inline]`, `[constexpr]`

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

### format\_no\_copy

`match_flag_type` `std::regex_constants::format_no_copy` `[inline]`, `[constexpr]`

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

**format\_sed**

`match_flag_type` `std::regex_constants::format_sed` [inline], [constexpr]

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX sed utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Referenced by `std::regex_traits< _CharT >::lookup_classname()`.

**grep**

`syntax_option_type` `std::regex_constants::grep` [inline], [constexpr]

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type basic`, except that newlines are treated as whitespace.

**icase**

`syntax_option_type` `std::regex_constants::icase` [inline], [constexpr]

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

**match\_any**

`match_flag_type` `std::regex_constants::match_any` [inline], [constexpr]

If more than one match is possible then any match is an acceptable result.

**match\_continuous**

`match_flag_type` `std::regex_constants::match_continuous` [inline], [constexpr]

The expression only matches a sub-sequence that begins at first .

Referenced by `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++()`.

**match\_default**

`match_flag_type` `std::regex_constants::match_default` [inline], [constexpr]

The default matching rules.

Referenced by `std::regex_iterator< const char * >::regex_iterator()`, `std::regex_token_iterator< const char * >::regex_token_iterator()`, `std::regex_token_iterator< const char * >::regex_token_iterator()`, `std::regex_token_iterator< const char * >::regex_token_iterator()`, `std::regex_token_iterator< const char * >::regex_token_iterator()`, `std::regex_match()`, `std::regex_match()`, `std::regex_match()`, `std::regex_match()`, `std::regex_match()`, `std::regex_match()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_search()`, `std::regex_search()`, `std::regex_search()`, `std::regex_search()`, and `std::regex_search()`.

**match\_not\_bol**

`match_flag_type` `std::regex_constants::match_not_bol` [inline], [constexpr]

The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

**match\_not\_bow**

`match_flag_type` `std::regex_constants::match_not_bow` [inline], [constexpr]

The expression `\b` is not matched against the sub-sequence [first,first).



**match\_not\_eol**

`match_flag_type` `std::regex_constants::match_not_eol` [inline], [constexpr]

The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character (\$) in the regular expression shall not match [last, last).

**match\_not\_eow**

`match_flag_type` `std::regex_constants::match_not_eow` [inline], [constexpr]

The expression \b should not be matched against the sub-sequence [last,last).

**match\_not\_null**

`match_flag_type` `std::regex_constants::match_not_null` [inline], [constexpr]

The expression does not match an empty sequence.

Referenced by `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator++()`.

**match\_prev\_avail**

`match_flag_type` `std::regex_constants::match_prev_avail` [inline], [constexpr]

--first is a valid iterator position. When this flag is set then the flags `match_not_bol` and `match_not_bow` are ignored by the algorithms `regex_match`, `regex_search`, and `regex_replace`, and by the iterators `regex_iterator` and `regex_token_iterator`.

Referenced by `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator++()`.

**multiline**

`syntax_option_type` `std::regex_constants::multiline` [inline], [constexpr]

Specifies that the ^ anchor matches at the beginning of a line, and the \$ anchor matches at the end of a line, not only at the beginning/end of the input. Valid for the ECMAScript syntax, ignored otherwise.

Since

C++17

**nosubs**

`syntax_option_type` `std::regex_constants::nosubs` [inline], [constexpr]

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied `match_results` structure.

**optimize**

`syntax_option_type` `std::regex_constants::optimize` [inline], [constexpr]

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

## 4.22 std::rel\_ops Namespace Reference

### Functions

- `template<class _Tp>`  
`bool operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`  
`bool operator<= (const _Tp &__x, const _Tp &__y)`

- `template<class _Tp>`  
`bool operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`  
`bool operator>= (const _Tp &__x, const _Tp &__y)`

#### 4.22.1 Detailed Description

The generated relational operators are sequestered here.

Libstdc++ headers must not use the contents of `rel_ops`. User code should also avoid them, because unconstrained function templates are too greedy and can easily cause ambiguities.

C++20 default comparisons are a better solution.

#### 4.22.2 Function Documentation

##### **operator"!="()**

```
template<class _Tp>
bool std::rel_ops::operator!= (
    const _Tp & __x,
    const _Tp & __y) [inline]
```

Defines != for arbitrary types, in terms of ==.

##### Parameters

|                              |                |
|------------------------------|----------------|
| $\longleftrightarrow$<br>__x | A thing.       |
| $\longleftrightarrow$<br>__y | Another thing. |

##### Returns

`__x != __y`

This function uses == to determine its result.

##### **operator<=()**

```
template<class _Tp>
bool std::rel_ops::operator<= (
    const _Tp & __x,
    const _Tp & __y) [inline]
```

Defines <= for arbitrary types, in terms of <.

##### Parameters

|                              |                |
|------------------------------|----------------|
| $\longleftrightarrow$<br>__x | A thing.       |
| $\longleftrightarrow$<br>__y | Another thing. |

##### Returns

`__x <= __y`

This function uses < to determine its result.

**operator>()**

```
template<class _Tp>
bool std::rel_ops::operator> (
    const _Tp & __x,
    const _Tp & __y) [inline]
```

Defines > for arbitrary types, in terms of <.

**Parameters**

|                              |                |
|------------------------------|----------------|
| $\longleftrightarrow$<br>__x | A thing.       |
| $\longleftrightarrow$<br>__y | Another thing. |

**Returns**

\_\_x > \_\_y

This function uses < to determine its result.

**operator>=()**

```
template<class _Tp>
bool std::rel_ops::operator>= (
    const _Tp & __x,
    const _Tp & __y) [inline]
```

Defines >= for arbitrary types, in terms of <.

**Parameters**

|                              |                |
|------------------------------|----------------|
| $\longleftrightarrow$<br>__x | A thing.       |
| $\longleftrightarrow$<br>__y | Another thing. |

**Returns**

\_\_x >= \_\_y

This function uses < to determine its result.

**4.23 std::this\_thread Namespace Reference****Functions**

- [thread::id get\\_id](#) () noexcept
- template<typename \_Rep, typename \_Period>  
void [sleep\\_for](#) (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration>  
void [sleep\\_until](#) (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void [yield](#) () noexcept

**4.23.1 Detailed Description**

ISO C++ 2011 namespace for interacting with the current thread.

C++11 30.3.2 [thread.thread.this] Namespace this\_thread.

### 4.23.2 Function Documentation

#### get\_id()

```
thread::id std::this_thread::get_id () [inline], [noexcept]
```

The unique identifier of the current thread.

#### sleep\_for()

```
template<typename _Rep, typename _Period>
void std::this_thread::sleep_for (
    const chrono::duration< _Rep, _Period > & __rtime) [inline]
```

this\_thread::sleep\_for

References [std::chrono::duration\\_cast\(\)](#).

Referenced by [sleep\\_until\(\)](#).

#### sleep\_until()

```
template<typename _Clock, typename _Duration>
void std::this_thread::sleep_until (
    const chrono::time_point< _Clock, _Duration > & __atime) [inline]
```

this\_thread::sleep\_until

References [sleep\\_for\(\)](#).

#### yield()

```
void std::this_thread::yield () [inline], [noexcept]
```

Allow the implementation to schedule a different thread.

## 4.24 std::tr1 Namespace Reference

### Namespaces

- namespace [\\_\\_detail](#)

### Functions

- template<typename \_Tp>  
[std::complex](#)< \_Tp > **acos** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tp>  
[std::complex](#)< \_Tp > **acosh** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tp>  
\_Tp **arg** (const [complex](#)< \_Tp > &)
- template<typename \_Tp>  
[std::complex](#)< \_Tp > **asin** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tp>  
[std::complex](#)< \_Tp > **asinh** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **assoc\_laguerre** (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- float **assoc\_laguerref** (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- long double **assoc\_laguerrel** (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **assoc\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- float **assoc\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- long double **assoc\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)

- `template<typename _Tp>`  
`std::complex< _Tp > atan (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > atanh (const std::complex< _Tp > &)`
- `template<typename _Tpx, typename _Tpy>`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type beta (_Tpx __x, _Tpy __y)`
- `float betaf (float __x, float __y)`
- `long double betal (long double __x, long double __y)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)`
- `float comp_ellint_1f (float __k)`
- `long double comp_ellint_1l (long double __k)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)`
- `float comp_ellint_2f (float __k)`
- `long double comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn>`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp_ellint_3 (_Tp __k, _Tpn __nu)`
- `float comp_ellint_3f (float __k, float __nu)`
- `long double comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float conf_hypergf (float __a, float __c, float __x)`
- `long double conf_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tp>`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > conj (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > conj (const std::complex< _Tp > &__z)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_if (float __nu, float __x)`
- `long double cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_jf (float __nu, float __x)`
- `long double cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_kf (float __nu, float __x)`
- `long double cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)`
- `float cyl_neumannf (float __nu, float __x)`
- `long double cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_1 (_Tp __k, _Tpp __phi)`
- `float ellint_1f (float __k, float __phi)`
- `long double ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_2 (_Tp __k, _Tpp __phi)`
- `float ellint_2f (float __k, float __phi)`
- `long double ellint_2l (long double __k, long double __phi)`

- `template<typename _Tp, typename _Tpn, typename _Tpp>`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi)`
- `float ellint_3f (float __k, float __nu, float __phi)`
- `long double ellint_3l (long double __k, long double __nu, long double __phi)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type expint ( _Tp __x)`
- `float expintf (float __x)`
- `long double expintl (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type fabs ( _Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > fabs (const std::complex< _Tp > &__z)`
- `float fabs (float __x)`
- `long double fabs (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)`
- `float hermitef (unsigned int __n, float __x)`
- `long double hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg ( _Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float hypergf (float __a, float __b, float __c, float __x)`
- `long double hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<typename _Tp>`  
`constexpr _Tp imag (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)`
- `float laguerref (unsigned int __n, float __x)`
- `long double laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __n, _Tp __x)`
- `float legendref (unsigned int __n, float __x)`
- `long double legendrel (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`_Tp constexpr norm (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > polar (const _Tp &, const _Tp &=_Tp(0))`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp, typename _Up>`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type pow ( _Tp __x, _Up __y)`
- `template<typename _Tp>`  
`std::complex< _Tp > pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const _Up &__y)`

- `template<typename _Tp>`  
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `float pow (float __x, float __y)`
- `long double pow (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr _Tp real (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __x)`
- `float riemann_zetaf (float __x)`
- `long double riemann_zetal (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)`
- `float sph_besself (unsigned int __n, float __x)`
- `long double sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float sph_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)`
- `float sph_neumannf (unsigned int __n, float __x)`
- `long double sph_neumannl (unsigned int __n, long double __x)`

#### 4.24.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is `std::tr1`.

#### 4.24.2 Function Documentation

##### **arg()**

```
template<typename _Tp>
_Tp std::arg (
    const complex< _Tp > & __z) [inline]
```

Return phase angle of `z`.

##### **conf\_hyperg()**

```
template<typename _Tpa, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (
    _Tpa __a,
    _Tpc __c,
    _Tp __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of real numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

The confluent hypergeometric function is defined by

$${}_1F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$

## Parameters

|                         |                              |
|-------------------------|------------------------------|
| $\leftrightarrow$<br>_a | The numeratorial parameter   |
| $\leftrightarrow$<br>_c | The denominatorial parameter |
| $\leftrightarrow$<br>_x | The argument                 |

**conf\_hypergf()**

```
float __gnu_cxx::conf_hypergf (
    float __a,
    float __c,
    float __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of float numeratorial parameter a, denominatorial parameter c, and argument x.

## See also

conf\_hyperg for details.

**conf\_hypergl()**

```
long double __gnu_cxx::conf_hypergl (
    long double __a,
    long double __c,
    long double __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of long double numeratorial parameter a, denominatorial parameter c, and argument x.

## See also

conf\_hyperg for details.

**hyperg()**

```
template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (
    _Tpa __a,
    _Tpb __b,
    _Tpc __c,
    _Tp __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of real numeratorial parameters a and b, denominatorial parameter c, and argument x.

The hypergeometric function is defined by

$${}_2F_1(a; b; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$



## Parameters

|                         |                                   |
|-------------------------|-----------------------------------|
| $\leftrightarrow$<br>_a | The first numeratorial parameter  |
| $\leftrightarrow$<br>_b | The second numeratorial parameter |
| $\leftrightarrow$<br>_c | The denominatorial parameter      |
| $\leftrightarrow$<br>_x | The argument                      |

**hypergf()**

```
float __gnu_cxx::hypergf (
    float __a,
    float __b,
    float __c,
    float __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of @ float numeratorial parameters a and b, denominatorial parameter c, and argument x.

**See also**

hyperg for details.

**hypergl()**

```
long double __gnu_cxx::hypergl (
    long double __a,
    long double __b,
    long double __c,
    long double __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of long double numeratorial parameters a and b, denominatorial parameter c, and argument x.

**See also**

hyperg for details.

**norm()**

```
template<typename _Tp>
_Tp constexpr std::norm (
    const complex< _Tp > & ) [constexpr]
```

Return z magnitude squared.

**polar()**

```
template<typename _Tp>
complex< _Tp > std::polar (
    const _Tp & __rho,
    const _Tp & __theta = _Tp(0)) [inline]
```

Return complex with magnitude *rho* and angle *theta*.

## 4.25 std::tr1::\_\_detail Namespace Reference

### 4.25.1 Detailed Description

Implementation details not part of the namespace std::tr1 interface.

## 4.26 std::tr2 Namespace Reference

### Namespaces

- namespace [\\_\\_detail](#)

### Classes

- struct [\\_\\_dynamic\\_bitset\\_base](#)
- struct [\\_\\_reflection\\_typelist](#)
- struct [\\_\\_reflection\\_typelist< \\_First, \\_Rest... >](#)
- struct [\\_\\_reflection\\_typelist<>](#)
- struct [bases](#)
- class [bool\\_set](#)
- struct [direct\\_bases](#)
- class [dynamic\\_bitset](#)

### Functions

- bool **certainly** ([bool\\_set](#) \_\_b)
- bool **contains** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- bool **equals** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- bool **is\_emptyset** ([bool\\_set](#) \_\_b)
- bool **is\_indeterminate** ([bool\\_set](#) \_\_b)
- bool **is\_singleton** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **operator!=** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator!=** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- [bool\\_set](#) **operator!=** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator&** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator&** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [dynamic\\_bitset](#)< \_WordT, \_Alloc > &\_\_x)
- [bool\\_set](#) **operator==** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator==** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [dynamic\\_bitset](#)< \_WordT, \_Alloc > &\_\_x)
- [bool\\_set](#) **operator^** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator^** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- [bool\\_set](#) **operator|** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator|** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- bool **possibly** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **set\_complement** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **set\_intersection** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **set\_intersection** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- [bool\\_set](#) **set\_intersection** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)

- `bool_set set_union (bool __s, bool_set __t)`
- `bool_set set_union (bool_set __s, bool __t)`
- `bool_set set_union (bool_set __s, bool_set __t)`
- `template<typename _WordT, typename _Alloc>  
bool operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>  
bool operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>  
bool operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>  
bool operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>  
dynamic_bitset< _WordT, _Alloc > operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>  
dynamic_bitset< _WordT, _Alloc > operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>  
dynamic_bitset< _WordT, _Alloc > operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>  
dynamic_bitset< _WordT, _Alloc > operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`

#### 4.26.1 Detailed Description

Namespace for non-standard "TR2" extensions.

### 4.27 std::tr2::\_\_detail Namespace Reference

#### 4.27.1 Detailed Description

Implementation details not part of the namespace std::tr2 interface.

## 5 Class Documentation

### 5.1 \_\_gnu\_parallel::\_\_accumulate\_binop\_reduct<\_BinOp> Struct Template Reference

```
#include <for_each_selectors.h>
```

#### Public Member Functions

- `__accumulate_binop_reduct (_BinOp &__b)`
- `template<typename _Result, typename _Addend>  
_Result operator() (const _Result &__x, const _Addend &__y)`

**Public Attributes**

- `_BinOp` & `__binop`

**5.1.1 Detailed Description**

```
template<typename _BinOp>
struct __gnu_parallel::__accumulate_binop_reduct<_BinOp>
```

General reduction, using a binary operator.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**5.2 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference**

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:

**Public Member Functions**

- `template<typename _Op>`  
`std::iterator_traits<_It>::value_type operator() (_Op __o, _It __i)`

**Public Attributes**

- `_It` `_M_finish_iterator`

**5.2.1 Detailed Description**

```
template<typename _It>
struct __gnu_parallel::__accumulate_selector<_It>
```

`std::accumulate()` selector.

## 5.2.2 Member Function Documentation

### operator>()

```
template<typename _It>
template<typename _Op>
std::iterator_traits< _It >::value_type __gnu_parallel::__accumulate_selector< _It >::operator()
(
    _Op __o,
    _It __i) [inline]
```

Functor execution.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator (unused).           |
| <code>__i</code> | iterator referencing object. |

#### Returns

The current value.

## 5.2.3 Member Data Documentation

### \_M\_finish\_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:
```

- [for\\_each\\_selectors.h](#)

## 5.3 \_\_gnu\_parallel::\_\_adjacent\_difference\_selector< \_It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector< _It >`:



**Public Member Functions**

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

**Public Attributes**

- `_It _M_finish_iterator`

**5.3.1 Detailed Description**

`template<typename _It>`  
`struct __gnu_parallel::__adjacent_difference_selector< _It >`

Selector that returns the difference between two adjacent `__elements`.

**5.3.2 Member Data Documentation****`_M_finish_iterator`**

`template<typename _It>`  
`_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator` [inherited]  
`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

The documentation for this struct was generated from the following file:

- `for_each_selectors.h`

**5.4 `__gnu_parallel::__adjacent_find_selector` Struct Reference**

`#include <find_selectors.h>`

Inheritance diagram for `__gnu_parallel::__adjacent_find_selector`:

**Public Member Functions**

- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

### 5.4.1 Detailed Description

Test predicate on two adjacent elements.

### 5.4.2 Member Function Documentation

#### **`_M_sequential_algorithm()`**

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm
(
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

References [std::make\\_pair\(\)](#).

#### **`operator()()`**

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__adjacent_find_selector::operator() (
    _RAIter1 __i1,
    _RAIter2 __i2,
    _Pred __pred) [inline]
```

Test on one position.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.5 `__gnu_cxx::__alloc_traits<_Alloc, typename >` Struct Template Reference

```
#include <alloc_traits.h>
```

Inheritance diagram for `__gnu_cxx::__alloc_traits<_Alloc, typename >`:



### Public Types

- typedef `std::allocator_traits<_Alloc> _Base_type`
- typedef `_Alloc allocator_type`
- typedef `_Base_type::const_pointer const_pointer`
- typedef `const value_type & const_reference`
- using `const_void_pointer`
- typedef `_Base_type::difference_type difference_type`
- using `is_always_equal`
- typedef `_Base_type::pointer pointer`
- using `propagate_on_container_copy_assignment`
- using `propagate_on_container_move_assignment`
- using `propagate_on_container_swap`
- template<typename `_Tp`>  
using `rebind_alloc`
- template<typename `_Tp`>  
using `rebind_traits`
- typedef `value_type & reference`
- typedef `_Base_type::size_type size_type`
- typedef `_Base_type::value_type value_type`
- using `void_pointer`

### Static Public Member Functions

- static constexpr bool `_S_always_equal ()`
- static constexpr bool `_S_nothrow_move ()`
- static constexpr void `_S_on_swap (_Alloc &__a, _Alloc &__b)`
- static constexpr bool `_S_propagate_on_copy_assign ()`
- static constexpr bool `_S_propagate_on_move_assign ()`
- static constexpr bool `_S_propagate_on_swap ()`
- static constexpr `_Alloc` `_S_select_on_copy (const _Alloc &__a)`
- static constexpr pointer `allocate (_Alloc &__a, size_type __n)`
- static constexpr pointer `allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)`



- static constexpr pointer `allocate` (`_Alloc &__a`, `size_type __n`)
- static constexpr pointer `allocate` (`_Alloc &__a`, `size_type __n`, `const_void_pointer __hint`)
- template<typename `_Ptr`, typename... `_Args`>  
static constexpr std::enable\_if\_t<\_\_is\_custom\_pointer< `_Ptr` >::value > **construct** (`_Alloc &__a`, `_Ptr __p`, `_Args &&... __args`) noexcept(noexcept(`_Base_type::construct(__a, std::__to_address(__p), std::forward<_Args>(__args)...)`)))
- template<typename `_Tp`, typename... `_Args`>  
requires `__can_construct<_Alloc, _Tp, _Args...>`  
static constexpr void **construct** (`_Alloc &__a`, `_Tp *__p`, `_Args &&... __args`) noexcept(`_S_nothrow_construct<_Tp, _Args...>()`)
- static constexpr void **deallocate** (`_Alloc &__a`, pointer `__p`, `size_type __n`)
- static constexpr void **deallocate** (`_Alloc &__a`, pointer `__p`, `size_type __n`)
- template<typename `_Ptr`>  
static constexpr std::enable\_if\_t<\_\_is\_custom\_pointer< `_Ptr` >::value > **destroy** (`_Alloc &__a`, `_Ptr __p`) noexcept(noexcept(`_Base_type::destroy(__a, std::__to_address(__p))`)))
- template<typename `_Tp`>  
static constexpr void **destroy** (`_Alloc &__a`, `_Tp *__p`) noexcept(`_S_nothrow_destroy<_Tp>()`)
- static constexpr `size_type` **max\_size** (const `_Alloc &__a`) noexcept
- static constexpr `_Alloc` **select\_on\_container\_copy\_construction** (const `_Alloc &__rhs`)

### 5.5.1 Detailed Description

```
template<typename _Alloc, typename = typename _Alloc::value_type>
struct __gnu_cxx::__alloc_traits<_Alloc, typename >
```

Uniform interface to C++98 and C++11 allocators.

### 5.5.2 Member Typedef Documentation

#### `const_void_pointer`

```
template<typename _Alloc>
using std::allocator_traits<_Alloc>::const_void_pointer [inherited]
```

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

#### `is_always_equal`

```
template<typename _Alloc>
using std::allocator_traits<_Alloc>::is_always_equal [inherited]
```

Whether all instances of the allocator type compare equal.

`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

#### `propagate_on_container_copy_assignment`

```
template<typename _Alloc>
using std::allocator_traits<_Alloc>::propagate_on_container_copy_assignment [inherited]
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

#### `propagate_on_container_move_assignment`

```
template<typename _Alloc>
using std::allocator_traits<_Alloc>::propagate_on_container_move_assignment [inherited]
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

#### **propagate\_on\_container\_swap**

```
template<typename _Alloc>
```

```
using std::allocator\_traits< _Alloc >::propagate_on_container_swap [inherited]
```

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

#### **void\_pointer**

```
template<typename _Alloc>
```

```
using std::allocator\_traits< _Alloc >::void_pointer [inherited]
```

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

### 5.5.3 Member Function Documentation

#### **allocate()** [1/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr pointer std::allocator\_traits< _Alloc >::allocate (
    _Alloc & __a,
    size_type __n) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

##### Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

#### **allocate()** [2/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr pointer std::allocator\_traits< _Alloc >::allocate (
    _Alloc & __a,
    size_type __n,
    const_void_pointer __hint) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

##### Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

##### Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

**allocate()** [3/4]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
    _Alloc & __a,
    size_type __n) [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory.

**Parameters**

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

**allocate()** [4/4]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
    _Alloc & __a,
    size_type __n,
    const_void_pointer __hint) [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

**Returns**

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

**construct()**

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp, typename... _Args>
requires __can_construct<_Alloc, _Tp, _Args...>
static constexpr void std::allocator_traits< _Alloc >::construct (
    _Alloc & __a,
    _Tp * __p,
    _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`

**Parameters**

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

**deallocate()** [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr void std::allocator_traits<_Alloc >::deallocate (
    _Alloc & __a,
    pointer __p,
    size_type __n) [inline], [static], [constexpr]
```

Deallocate memory.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__a</code> | An allocator.                                  |
| <code>__p</code> | Pointer to the memory to deallocate.           |
| <code>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

**deallocate()** [2/2]

```
template<typename _Alloc>
static constexpr void std::allocator_traits<_Alloc >::deallocate (
    _Alloc & __a,
    pointer __p,
    size_type __n) [inline], [static], [constexpr], [inherited]
```

Deallocate memory.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__a</code> | An allocator.                                  |
| <code>__p</code> | Pointer to the memory to deallocate.           |
| <code>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

**destroy()**

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp>
static constexpr void std::allocator_traits<_Alloc >::destroy (
    _Alloc & __a,
    _Tp * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Tp`.

## Parameters

|       |                                  |
|-------|----------------------------------|
| $\_a$ | An allocator.                    |
| $\_p$ | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~Tp()`

**max\_size()**

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr size_type std::allocator\_traits<_Alloc>::max_size (
    const _Alloc & __a) [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

## Parameters

|       |               |
|-------|---------------|
| $\_a$ | An allocator. |
|-------|---------------|

## Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

**select\_on\_container\_copy\_construction()**

```
template<typename _Alloc>
static constexpr _Alloc std::allocator\_traits<_Alloc>::select_on_container_copy_construction (
    const _Alloc & __rhs) [inline], [static], [constexpr], [inherited]
```

Obtain an allocator to use when copying a container.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

## Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

The documentation for this struct was generated from the following file:

- [ext/alloc\\_traits.h](#)

**5.6 std::\_\_basic\_future<\_Res> Class Template Reference**

```
#include <future>
```

Inheritance diagram for std::\_\_basic\_future< \_Res >:



### Public Member Functions

- `__basic_future` (const `__basic_future` &)=delete
- `__basic_future` & `operator=` (const `__basic_future` &)=delete
- bool `valid` () const noexcept
- void `wait` () const
- template<typename \_Rep, typename \_Period>  
`future_status wait_for` (const `chrono::duration`< \_Rep, \_Period > &\_\_rel) const
- template<typename \_Clock, typename \_Duration>  
`future_status wait_until` (const `chrono::time_point`< \_Clock, \_Duration > &\_\_abs) const

### Protected Types

- typedef `__future_base::Result`< \_Res > & `__result_type`
- typedef `shared_ptr`< \_State\_base > `__state_type`

### Protected Member Functions

- `__basic_future` (const `__state_type` &\_\_state)
- `__basic_future` (const `shared_future`< \_Res > &) noexcept
- `__basic_future` (`future`< \_Res > &&) noexcept
- `__basic_future` (`shared_future`< \_Res > &&) noexcept
- `__result_type` `_M_get_result` () const
- void `_M_swap` (`__basic_future` &\_\_that) noexcept

#### 5.6.1 Detailed Description

```
template<typename _Res>
class std::__basic_future< _Res >
```

Common implementation for future and shared\_future.

## 5.6.2 Member Function Documentation

### `_M_get_result()`

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

The documentation for this class was generated from the following file:

- [future](#)

## 5.7 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

```
#include <base.h>
```

Inheritance diagram for `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



### Public Types

- typedef `_SecondArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

### Public Member Functions

- `__binder1st` (const `_Operation` &\_\_x, const `_FirstArgumentType` &\_\_y)
- `_ResultType operator()` (`_SecondArgumentType` &\_\_x) const
- `_ResultType operator()` (const `_SecondArgumentType` &\_\_x)

### Protected Attributes

- `_Operation` `_M_op`
- `_FirstArgumentType` `_M_value`

### 5.7.1 Detailed Description

`template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>`

`class __gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>`

Similar to `std::binder1st`, but giving the argument types explicitly.

### 5.7.2 Member Typedef Documentation

#### `argument_type`

`typedef _SecondArgumentType std::unary_function<_SecondArgumentType, _ResultType>::argument_type [inherited]`

`argument_type` is the type of the argument

#### `result_type`

`typedef _ResultType std::unary_function<_SecondArgumentType, _ResultType>::result_type [inherited]`

`result_type` is the return type

The documentation for this class was generated from the following file:

- [base.h](#)

## 5.8 `__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>` Class Template Reference

`#include <base.h>`

Inheritance diagram for `__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>`:



### Public Types

- `typedef _FirstArgumentType` [argument\\_type](#)
- `typedef _ResultType` [result\\_type](#)



## Public Member Functions

- **\_\_binder2nd** (const `_Operation` &\_\_x, const `_SecondArgumentType` &\_\_y)
- `_ResultType` **operator()** (`_FirstArgumentType` &\_\_x)
- `_ResultType` **operator()** (const `_FirstArgumentType` &\_\_x) const

## Protected Attributes

- `_Operation` **\_M\_op**
- `_SecondArgumentType` **\_M\_value**

### 5.8.1 Detailed Description

template<typename `_Operation`, typename `_FirstArgumentType`, typename `_SecondArgumentType`, typename `_ResultType`>

class `__gnu_parallel::__binder2nd`< `_Operation`, `_FirstArgumentType`, `_SecondArgumentType`, `_ResultType` >

Similar to `std::binder2nd`, but giving the argument types explicitly.

### 5.8.2 Member Typedef Documentation

#### argument\_type

typedef `_FirstArgumentType` `std::unary_function`< `_FirstArgumentType`, `_ResultType` >::argument\_type [inherited]

`argument_type` is the type of the argument

#### result\_type

typedef `_ResultType` `std::unary_function`< `_FirstArgumentType`, `_ResultType` >::result\_type [inherited]

`result_type` is the return type

The documentation for this class was generated from the following file:

- [base.h](#)

## 5.9 `std::__codecvt_abstract_base`< `_InternT`, `_ExternT`, `_StateT` > Class Template Reference

#include <codecvt.h>

Inheritance diagram for `std::__codecvt_abstract_base`< `_InternT`, `_ExternT`, `_StateT` >:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

### Public Member Functions

- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Protected Member Functions

- **\_\_codecvt\_abstract\_base** (size\_t \_\_refs=0)
- virtual bool **do\_always\_noconv** () const throw () =0
- virtual int **do\_encoding** () const throw () =0
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const =0
- virtual int **do\_max\_length** () const throw () =0
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const =0
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const =0

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_type\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 5.9.1 Detailed Description

template<typename \_InternT, typename \_ExternT, typename \_StateT>

class std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_StateT >

Common base for codecvt functions.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

#### 5.9.2 Member Function Documentation

##### do\_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type * __from_next,
```

```
extern_type * __to,
extern_type * __to_end,
extern_type *& __to_next) const [protected], [pure virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implemented in `std::codecvt< _InternT, _ExternT, _StateT >`, `std::codecvt< _Elem, char, mbstate_t >`, `std::codecvt< _InternT, _ExternT, std::codecvt< _InternT, _ExternT, encoding_state >, std::codecvt< char, char, mbstate_t >, std::codecvt< char, char, mbstate_t >, std::codecvt< char16_t, char, mbstate_t >, std::codecvt< char16_t, char, mbstate_t >, std::codecvt< char32_t, char, mbstate_t >, std::codecvt< char32_t, char, mbstate_t >, std::codecvt< wchar_t, char, mbstate_t >, and std::codecvt< wchar_t, char, mbstate_t >`. Referenced by `out()`.

**in()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

Returns

`codecvt_base::result`.

**out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

References [do\\_out\(\)](#).

**unshift()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types. The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.10 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread >` Struct Template Reference

```
#include <mt_allocator.h>
```

### 5.10.1 Detailed Description

```
template<template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__common_pool_policy<_PoolTp, _Thread >
```

Policy for shared `__pool` objects.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.11 `__gnu_parallel::__count_if_selector<_It, _Diff >` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__count_if_selector<_It, _Diff >`:



**Public Member Functions**

- `template<typename _Op>`  
`_Diff operator() (_Op &__o, _It __i)`

**Public Attributes**

- `_It _M_finish_iterator`

**5.11.1 Detailed Description**

`template<typename _It, typename _Diff>`  
`struct __gnu_parallel::__count_if_selector<_It, _Diff>`

`std::count_if()` selector.

**5.11.2 Member Function Documentation****`operator()()`**

```
template<typename _It, typename _Diff>
template<typename _Op>
_Diff __gnu_parallel::__count_if_selector<_It, _Diff>::operator() (
    _Op & __o,
    _It __i) [inline]
```

Functor execution.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator.                    |
| <code>__i</code> | iterator referencing object. |

**Returns**

1 if count, 0 if does not count.

**5.11.3 Member Data Documentation****`_M_finish_iterator`**

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**5.12 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference**

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__count_selector<_It, _Diff>`:



### Public Member Functions

- `template<typename _ValueType>`  
`_Diff operator() (_ValueType &__v, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.12.1 Detailed Description

```
template<typename _It, typename _Diff>
struct __gnu_parallel::__count_selector<_It, _Diff>
```

`std::count()` selector.

#### 5.12.2 Member Function Documentation

##### `operator()()`

```
template<typename _It, typename _Diff>
template<typename _ValueType>
_Diff __gnu_parallel::__count_selector<_It, _Diff>::operator() (
    _ValueType & __v,
    _It __i) [inline]
```

Functor execution.

##### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__v</code> | Current value.               |
| <code>__i</code> | iterator referencing object. |

## Returns

1 if count, 0 if does not count.

## 5.12.3 Member Data Documentation

**\_M\_finish\_iterator**

```
template<typename _It>
```

```
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

\_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.13 std::\_\_ctype\_abstract\_base&lt;\_CharT&gt; Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::\_\_ctype\_abstract\_base<\_CharT>:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- const **char\_type** \* **is** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, mask \* \_\_vec) const
- bool **is** (mask \_\_m, **char\_type** \_\_c) const
- char **narrow** (**char\_type** \_\_c, char \_\_dfault) const
- const **char\_type** \* **narrow** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- const **char\_type** \* **scan\_is** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const



- const `char_type` \* `scan_not` (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const
- const `char_type` \* `tolower` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const
- `char_type` `tolower` (`char_type` `__c`) const
- const `char_type` \* `toupper` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const
- `char_type` `toupper` (`char_type` `__c`) const
- `char_type` `widen` (char `__c`) const
- const char \* `widen` (const char \* `__lo`, const char \* `__hi`, `char_type` \* `__to`) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- `__ctype_abstract_base` (size\_t `__refs=0`)
- virtual const `char_type` \* `do_is` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, mask \* `__vec`) const =0
- virtual bool `do_is` (mask `__m`, `char_type` `__c`) const =0
- virtual char `do_narrow` (`char_type` `__c`, char `__default`) const =0
- virtual const `char_type` \* `do_narrow` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, char `__default`, char \* `__to`) const =0
- virtual const `char_type` \* `do_scan_is` (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const =0
- virtual const `char_type` \* `do_scan_not` (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const =0
- virtual const `char_type` \* `do_tolower` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const =0
- virtual `char_type` `do_tolower` (`char_type` `__c`) const =0
- virtual const `char_type` \* `do_toupper` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const =0
- virtual `char_type` `do_toupper` (`char_type` `__c`) const =0
- virtual `char_type` `do_widen` (char `__c`) const =0
- virtual const char \* `do_widen` (const char \* `__lo`, const char \* `__hi`, `char_type` \* `__to`) const =0

### Static Protected Member Functions

- static `__c_locale` `_S_clone_c_locale` (`__c_locale` & `__cloc`) throw ()
- static void `_S_create_c_locale` (`__c_locale` & `__cloc`, const char \* `__s`, `__c_locale` `__old=0`)
- static void `_S_destroy_c_locale` (`__c_locale` & `__cloc`)
- static `__c_locale` `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` `__cloc`, const char \* `__s`)

### 5.13.1 Detailed Description

**template<typename \_CharT>**  
**class std::\_\_ctype\_abstract\_base<\_CharT>**

Common base for ctype facet.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

### 5.13.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT>
typedef _CharT std::__ctype_abstract_base<_CharT>::char_type
```

Typedef for the template parameter.

### 5.13.3 Member Function Documentation

#### do\_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base<_CharT>::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [protected], [pure virtual]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

Implemented in `std::ctype<_CharT>`, `std::ctype<wchar_t>`, `std::ctype<wchar_t>`, `std::mask<_CharT>`, and `std::mask<_CharT>`.

#### do\_is() [2/2]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base<_CharT>::do_is (
    mask __m,
    char_type __c) const [protected], [pure virtual]
```

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

|                                   |                                                 |
|-----------------------------------|-------------------------------------------------|
| <a href="#"><code>_↔_c</code></a> | The <code>char_type</code> to find the mask of. |
| <a href="#"><code>_↔_m</code></a> | The mask to compare against.                    |

**Returns**

`(M & __m) != 0.`

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), [std::ctype<wchar\\_t>](#), [std::mask<\\_CharT>](#), and [std::mask<\\_CharT>](#).

Referenced by [std::mask<\\_CharT>::is\(\)](#), and [std::mask<\\_CharT>::is\(\)](#).

**do\_narrow()** [1/2]

```
template<typename _CharT>
virtual char std::\_\_ctype\_abstract\_base<\_CharT>::do\_narrow (
    char_type __c,
    char __dfault) const [protected], [pure virtual]
```

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                       |                                        |
|-----------------------|----------------------------------------|
| <code>__c</code>      | The <code>char_type</code> to convert. |
| <code>__dfault</code> | Char to return if conversion fails.    |

**Returns**

The converted `char`.

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), [std::ctype<wchar\\_t>](#), [std::mask<\\_CharT>](#), and [std::mask<\\_CharT>](#).

Referenced by [std::mask<\\_CharT>::narrow\(\)](#), and [std::mask<\\_CharT>::narrow\(\)](#).

**do\_narrow()** [2/2]

```
template<typename _CharT>
virtual const char_type * std::\_\_ctype\_abstract\_base<\_CharT>::do\_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __dfault,
    char * __to) const [protected], [pure virtual]
```

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

## Returns

`__hi`.

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), [std::ctype<wchar\\_t>](#), [std::mask<\\_CharT>](#), and [std::mask<\\_CharT>](#).

**do\_scan\_is()**

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base<_CharT>::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [pure virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a matching char\_type if found, else `__hi`.

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), [std::ctype<wchar\\_t>](#), [std::mask<\\_CharT>](#), and [std::mask<\\_CharT>](#).

Referenced by [std::mask<\\_CharT>::scan\\_is\(\)](#).

**do\_scan\_not()**

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base<_CharT>::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [pure virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [`lo`,`hi`) for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

|                                                                                                                        |                              |
|------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a non-matching `char_type` if found, else `__hi`.

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), [std::ctype<wchar\\_t>](#), [std::mask<\\_CharT>](#), and [std::mask<\\_CharT>](#).

Referenced by [std::mask<\\_CharT>::scan\\_not\(\)](#).

**do\_tolower()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base<_CharT>::do_tolower (
    char_type * __lo,
    const char_type * __hi) const [protected], [pure virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), [std::ctype<wchar\\_t>](#), [std::mask<\\_CharT>](#), and [std::mask<\\_CharT>](#).

**do\_tolower()** [2/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base<_CharT>::do_tolower (
    char_type __c) const [protected], [pure virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

|                         |                           |
|-------------------------|---------------------------|
| $\leftrightarrow$<br>_c | The char_type to convert. |
|-------------------------|---------------------------|

## Returns

The lowercase char\_type if convertible, else \_\_c.

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), [std::ctype<wchar\\_t>](#), [std::mask<\\_CharT>](#), and [std::mask<\\_CharT>](#).

Referenced by [std::mask<\\_CharT>::tolower\(\)](#), and [std::mask<\\_CharT>::tolower\(\)](#).

**do\_toupper()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base<_CharT>::do_toupper (
    char_type * __lo,
    const char_type * __hi) const [protected], [pure virtual]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

|                          |                            |
|--------------------------|----------------------------|
| $\leftrightarrow$<br>_lo | Pointer to start of range. |
| $\leftrightarrow$<br>_hi | Pointer to end of range.   |

## Returns

`__hi`.

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), [std::ctype<wchar\\_t>](#), [std::mask<\\_CharT>](#), and [std::mask<\\_CharT>](#).

**do\_toupper()** [2/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base<_CharT>::do_toupper (
    char_type __c) const [protected], [pure virtual]
```

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

|                         |                           |
|-------------------------|---------------------------|
| $\leftrightarrow$<br>_c | The char_type to convert. |
|-------------------------|---------------------------|

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Implemented in `std::ctype< _CharT >`, `std::ctype< wchar_t >`, `std::ctype< wchar_t >`, `std::mask< _CharT >`, and `std::mask< _CharT >`.

Referenced by `std::mask< _CharT >::toupper()`, and `std::mask< _CharT >::toupper()`.

**do\_widen() [1/2]**

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_widen (
    char __c) const [protected], [pure virtual]
```

Widen char.

This virtual function converts the char to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The converted `char_type`

Implemented in `std::ctype< _CharT >`, `std::ctype< char >`, `std::ctype< char >`, `std::ctype< wchar_t >`, `std::ctype< wchar_t >`, `std::mask< _CharT >`, and `std::mask< _CharT >`.

Referenced by `std::mask< _CharT >::widen()`, and `std::mask< _CharT >::widen()`.

**do\_widen() [2/2]**

```
template<typename _CharT>
virtual const char * std::__ctype_abstract_base< _CharT >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [protected], [pure virtual]
```

Widen char array.

This function converts each char in the input to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start range.           |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

## Returns

`__hi`.

Implemented in `std::ctype<_CharT>`, `std::ctype<wchar_t>`, `std::ctype<wchar_t>`, `std::mask<_CharT>`, and `std::mask<_CharT>`.

**is()** [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

## Returns

`__hi`.**is()** [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base<_CharT>::is (
    mask __m,
    char_type __c) const [inline]
```

Test `char_type` classification.

This function finds a mask `M` for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_↵type>::do_is()`.

## Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>↵_c</code> | The <code>char_type</code> to compare the mask of. |
| <code>↵_m</code> | The mask to compare against.                       |

## Returns

`(M & __m) != 0.`

Referenced by `std::time_get<_CharT, _InIter>::get()`.



**narrow()** [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
    char_type __c,
    char __default) const [inline]
```

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `default` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

**Returns**

The converted `char`.

Referenced by `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_get< _CharT, _InIter >::get()`, and `std::time_put< _CharT, _OutIter >::put()`.

**narrow()** [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [inline]
```

Narrow array to `char` array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `default` is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**scan\_is()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
    mask __m,
```

```
const char_type * __lo,
const char_type * __hi) const [inline]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

#### Parameters

|                                                                                                     |                              |
|-----------------------------------------------------------------------------------------------------|------------------------------|
|  <code>__m</code>  | The mask to compare against. |
|  <code>__lo</code> | Pointer to start of range.   |
|  <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to matching char\_type if found, else `__hi`.

#### scan\_not()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

#### Parameters

|                                                                                                       |                                 |
|-------------------------------------------------------------------------------------------------------|---------------------------------|
|  <code>__m</code>  | The mask to compare against.    |
|  <code>__lo</code> | Pointer to first char in range. |
|  <code>__hi</code> | Pointer to end of range.        |

#### Returns

Pointer to non-matching char if found, else `__hi`.

#### tolower() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::tolower (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

|                                               |                            |
|-----------------------------------------------|----------------------------|
| <a href="#"><u>↔</u></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"><u>↔</u></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**tolower()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
    char_type __c) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                                              |                                        |
|----------------------------------------------|----------------------------------------|
| <a href="#"><u>↔</u></a><br><code>__c</code> | The <code>char_type</code> to convert. |
|----------------------------------------------|----------------------------------------|

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Referenced by `std::time_get< _CharT, _InIter >::get()`.

**toupper()** [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

**Parameters**

|                                               |                            |
|-----------------------------------------------|----------------------------|
| <a href="#"><u>↔</u></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"><u>↔</u></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**toupper()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::toupper (
    char_type __c) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else `__c`.

Referenced by `std::time_get<_CharT, _Inlter>::get()`.

**widen()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::widen (
    char __c) const [inline]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The converted char\_type.

Referenced by `std::money_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get()`, `std::money_put<_CharT, _Outlter>::do_put()`, and `std::time_put<_CharT, _Outlter>::do_put()`.

**widen()** [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base<_CharT>::widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [inline]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                                   |                                   |
|-----------------------------------|-----------------------------------|
| <a href="#"><code>__lo</code></a> | Pointer to start of range.        |
| <a href="#"><code>__hi</code></a> | Pointer to end of range.          |
| <a href="#"><code>__to</code></a> | Pointer to the destination array. |

## Returns

[`\_\_hi`](#).

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.14 `std::tr2::__dynamic_bitset_base<_WordT, _Alloc>` Struct Template Reference

```
#include <dynamic_bitset>
```

Inheritance diagram for `std::tr2::__dynamic_bitset_base<_WordT, _Alloc>`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_WordT` **block\_type**
- typedef `size_t` **size\_type**

## Public Member Functions

- `__dynamic_bitset_base` ([`\_\_dynamic\_bitset\_base`](#) &&\_\_b)=default
- `__dynamic_bitset_base` (const [`\_\_dynamic\_bitset\_base`](#) &)=default
- `__dynamic_bitset_base` (const allocator\_type &\_\_alloc)
- `__dynamic_bitset_base` (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type &[`\_\_alloc`](#)=allocator\_type())

- `size_t _M_are_all_aux ()` const noexcept
- `void _M_clear ()` noexcept
- `void _M_do_and (const __dynamic_bitset_base &__x)` noexcept
- `void _M_do_append_block (block_type __block, size_type __pos)`
- `size_t _M_do_count ()` const noexcept
- `void _M_do_dif (const __dynamic_bitset_base &__x)` noexcept
- `size_type _M_do_find_first (size_t __not_found)` const
- `size_type _M_do_find_next (size_t __prev, size_t __not_found)` const
- `void _M_do_flip ()` noexcept
- `void _M_do_left_shift (size_t __shift)`
- `void _M_do_or (const __dynamic_bitset_base &__x)` noexcept
- `void _M_do_reset ()` noexcept
- `void _M_do_right_shift (size_t __shift)`
- `void _M_do_set ()` noexcept
- `unsigned long long _M_do_to_ullong ()` const
- `unsigned long _M_do_to_ulong ()` const
- `void _M_do_xor (const __dynamic_bitset_base &__x)` noexcept
- `allocator_type _M_get_allocator ()` const noexcept
- `block_type _M_getword (size_type __pos)` const noexcept
- `block_type & _M_getword (size_type __pos)` noexcept
- `block_type _M_hiword ()` const noexcept
- `block_type & _M_hiword ()` noexcept
- `bool _M_is_any ()` const noexcept
- `bool _M_is_equal (const __dynamic_bitset_base &__x)` const noexcept
- `bool _M_is_less (const __dynamic_bitset_base &__x)` const noexcept
- `bool _M_is_proper_subset_of (const __dynamic_bitset_base &__b)` const noexcept
- `bool _M_is_subset_of (const __dynamic_bitset_base &__b)` noexcept
- `void _M_resize (size_t __nbits, bool __value)`
- `size_type _M_size ()` const noexcept
- `void _M_swap (__dynamic_bitset_base &__b)` noexcept
- `__dynamic_bitset_base & operator= (__dynamic_bitset_base &&)=default`
- `__dynamic_bitset_base & operator= (const __dynamic_bitset_base &)=default`

#### Static Public Member Functions

- `static block_type _S_maskbit (size_type __pos)` noexcept
- `static size_type _S_whichbit (size_type __pos)` noexcept
- `static size_type _S_whichbyte (size_type __pos)` noexcept
- `static size_type _S_whichword (size_type __pos)` noexcept

#### Public Attributes

- `std::vector< block_type, allocator_type > _M_w`

#### Static Public Attributes

- `static const size_type _S_bits_per_block`
- `static const size_type npos`

### 5.14.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >
```

Base class, general case.

See documentation for `dynamic_bitset`.

### 5.14.2 Member Data Documentation

#### `_M_w`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::vector<block_type, allocator_type> std::tr2::__dynamic_bitset_base< _WordT, _Alloc >::_M_w
0 is the least significant word.
```

The documentation for this struct was generated from the following files:

- [dynamic\\_bitset](#)
- [dynamic\\_bitset.tcc](#)

## 5.15 `__gnu_parallel::__fill_selector<_It>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__fill_selector<_It>`:



### Public Member Functions

- `template<typename _ValueType>`  
`bool operator() (_ValueType &__v, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

### 5.15.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__fill_selector<_It>
```

`std::fill()` selector.

### 5.15.2 Member Function Documentation

#### `operator()()`

```
template<typename _It>
template<typename _ValueType>
bool __gnu_parallel::__fill_selector<_It>::operator() (
    _ValueType & __v,
    _It __i) [inline]
```

Functor execution.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__v</code> | Current value.               |
| <code>__i</code> | iterator referencing object. |

### 5.15.3 Member Data Documentation

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

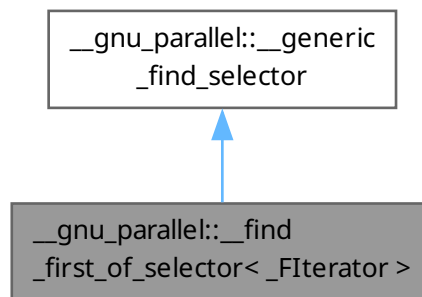
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.16 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference

```
#include <find_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__find_first_of_selector<_FIterator>`:



#### Public Member Functions

- `__find_first_of_selector` (`_FIterator __begin`, `_FIterator __end`)



- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

## Public Attributes

- `_FIterator \_M\_begin`
- `_FIterator \_M\_end`

### 5.16.1 Detailed Description

`template<typename _FIterator>`  
`struct \_\_gnu\_parallel::\_\_find\_first\_of\_selector< _FIterator >`

Test predicate on several elements.

### 5.16.2 Member Function Documentation

#### `\_M\_sequential\_algorithm()`

```
template<typename _FIterator>
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > \_\_gnu\_parallel::\_\_find\_first\_of\_selector< _FIterator >::__M_sequential_algorithm (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

References [std::make\\_pair\(\)](#).

#### `operator()()`

```
template<typename _FIterator>
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool \_\_gnu\_parallel::\_\_find\_first\_of\_selector< _FIterator >::operator() (
    _RAIter1 __i1,
    _RAIter2 __i2,
    _Pred __pred) [inline]
```

Test on one position.

#### Parameters

|                   |                                        |
|-------------------|----------------------------------------|
| <code>__i1</code> | _Iterator on first sequence.           |
| <code>__i2</code> | _Iterator on second sequence (unused). |

## Parameters

|                     |                 |
|---------------------|-----------------|
| <code>__pred</code> | Find predicate. |
|---------------------|-----------------|

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.17 \_\_gnu\_parallel::\_\_find\_if\_selector Struct Reference

```
#include <find_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_find\_if\_selector:



## Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

## 5.17.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

## 5.17.2 Member Function Documentation

`_M_sequential_algorithm()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_if_selector::_M_sequential_algorithm (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

## Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

References [std::make\\_pair\(\)](#).

**operator()()**

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__find_if_selector::operator() (
    _RAIter1 __i1,
    _RAIter2 __i2,
    _Pred __pred) [inline]
```

Test on one position.

## Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

**5.18 \_\_gnu\_parallel::\_\_for\_each\_selector<\_It> Struct Template Reference**

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__for_each_selector<_It>`:

**Public Member Functions**

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

**Public Attributes**

- [\\_lt \\_M\\_finish\\_iterator](#)

**5.18.1 Detailed Description**

```
template<typename _It>
struct __gnu_parallel::__for_each_selector< _It >
```

std::for\_each() selector.

**5.18.2 Member Function Documentation****operator()()**

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__for_each_selector< _It >::operator() (
    _Op & __o,
    _It __i) [inline]
```

Functor execution.

**Parameters**

|                                      |                              |
|--------------------------------------|------------------------------|
| <a href="#">↩</a><br><code>_o</code> | Operator.                    |
| <a href="#">↩</a><br><code>_i</code> | iterator referencing object. |

**5.18.3 Member Data Documentation****M\_finish\_iterator**

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::__M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**5.19 \_\_cxxabiv1::\_\_forced\_unwind Class Reference**

```
#include <cxxabi_forced.h>
```

**5.19.1 Detailed Description**

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

The documentation for this class was generated from the following file:

- [cxxabi\\_forced.h](#)

## 5.20 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.20.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__generate_selector<_It>
```

`std::generate()` selector.

#### 5.20.2 Member Function Documentation

##### `operator()()`

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__generate_selector<_It>::operator() (
    _Op & __o,
    _It __i) [inline]
```

Functor execution.

##### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator.                    |
| <code>__i</code> | iterator referencing object. |

### 5.20.3 Member Data Documentation

#### M\_finish\_iterator

```
template<typename _It>
```

```
_It __gnu_parallel::__generic_for_each_selector< _It >::__M_finish_iterator [inherited]  
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.21 \_\_gnu\_parallel::\_\_generic\_find\_selector Struct Reference

```
#include <find_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_generic\_find\_selector:



### 5.21.1 Detailed Description

Base class of all \_\_gnu\_parallel::\_\_find\_template selectors.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.22 \_\_gnu\_parallel::\_\_generic\_for\_each\_selector<\_It> Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__generic_for_each_selector<_It>`:



#### Public Attributes

- `_It` [M\\_finish\\_iterator](#)

### 5.22.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__generic_for_each_selector<_It>
```

Generic `__selector` for embarrassingly parallel functions.

### 5.22.2 Member Data Documentation

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:
```

- [for\\_each\\_selectors.h](#)

## 5.23 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__identity_selector<_It>`:



### Public Member Functions

- `template<typename _Op>`  
`_It operator() (_Op __o, _It __i)`

### Public Attributes

- `_It` [\\_M\\_finish\\_iterator](#)

### 5.23.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__identity_selector<_It>
```

Selector that just returns the passed iterator.



### 5.23.2 Member Function Documentation

#### operator>()

```
template<typename _It>
template<typename _Op>
_It __gnu_parallel::__identity_selector< _It >::operator() (
    _Op __o,
    _It __i) [inline]
```

Functor execution.

#### Parameters

|                    |                              |
|--------------------|------------------------------|
| $\leftarrow$<br>_o | Operator (unused).           |
| $\leftarrow$<br>_i | iterator referencing object. |

#### Returns

Passed iterator.

### 5.23.3 Member Data Documentation

#### \_M\_finish\_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:
```

- [for\\_each\\_selectors.h](#)

## 5.24 \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp >:



**Public Member Functions**

- [`\_\_inner\_product\_selector`](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op>`  
`_Tp operator()` (`_Op __mult, _It __current`)

**Public Attributes**

- `_It __begin1_iterator`
- `_It2 __begin2_iterator`
- `_It __M_finish_iterator`

**5.24.1 Detailed Description**

`template<typename _It, typename _It2, typename _Tp>`  
`struct __gnu_parallel::__inner_product_selector<_It,_It2,_Tp>`

`std::inner_product()` selector.

**5.24.2 Constructor & Destructor Documentation****`__inner_product_selector()`**

```
template<typename _It, typename _It2, typename _Tp>
__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__inner_product_selector (
    _It __b1,
    _It2 __b2) [inline], [explicit]
```

Constructor.

**Parameters**

|                   |                                    |
|-------------------|------------------------------------|
| <code>__b1</code> | Begin iterator of first sequence.  |
| <code>__b2</code> | Begin iterator of second sequence. |

References [`\_\_begin1\_iterator`](#), and [`\_\_begin2\_iterator`](#).

**5.24.3 Member Function Documentation****`operator>()()`**

```
template<typename _It, typename _It2, typename _Tp>
template<typename _Op>
_Tp __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator() (
    _Op __mult,
    _It __current) [inline]
```

Functor execution.

**Parameters**

|                        |                              |
|------------------------|------------------------------|
| <code>__mult</code>    | Multiplication functor.      |
| <code>__current</code> | iterator referencing object. |

**Returns**

Inner product elemental `__result`.

References [`\_\_begin1\_iterator`](#), and [`\_\_begin2\_iterator`](#).

#### 5.24.4 Member Data Documentation

##### `__begin1_iterator`

```
template<typename _It, typename _It2, typename _Tp>
    _It __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin1_iterator
```

Begin iterator of first sequence.

Referenced by `__inner_product_selector()`, and `operator()`.

##### `__begin2_iterator`

```
template<typename _It, typename _It2, typename _Tp>
    _It2 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin2_iterator
```

Begin iterator of second sequence.

Referenced by `__inner_product_selector()`, and `operator()`.

##### `_M_finish_iterator`

```
template<typename _It>
    _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.25 `std::__is_fast_hash<_Hash>` Struct Template Reference

```
#include <functional_hash.h>
```

#### 5.25.1 Detailed Description

```
template<typename _Hash>
struct std::__is_fast_hash< _Hash >
```

Hint about performance of hash functions.

If a given hash function object is not fast, the hash-based containers will cache the hash code. The default behavior is to consider that hashers are fast unless specified otherwise.

Users can specialize this for their own hash functions in order to force caching of hash codes in unordered containers.

Specializing this trait affects the ABI of the unordered containers, so use it carefully.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.26 `std::__is_location_invariant<_Tp>` Struct Template Reference

```
#include <std_function.h>
```

#### 5.26.1 Detailed Description

```
template<typename _Tp>
struct std::__is_location_invariant< _Tp >
```

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape.

Trivially copyable types are location-invariant and users can specialize this trait for other types.

The documentation for this struct was generated from the following file:

- [std\\_function.h](#)

## 5.27 `__gnu_parallel::__max_element_reduct<_Compare, _It>` Struct Template Reference

```
#include <for_each_selectors.h>
```

### Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

### Public Attributes

- `_Compare & __comp`

#### 5.27.1 Detailed Description

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__max_element_reduct<_Compare, _It>
```

Reduction for finding the maximum element, using a comparator.  
 The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.28 `__gnu_parallel::__min_element_reduct<_Compare, _It>` Struct Template Reference

```
#include <for_each_selectors.h>
```

### Public Member Functions

- `__min_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

### Public Attributes

- `_Compare & __comp`

#### 5.28.1 Detailed Description

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__min_element_reduct<_Compare, _It>
```

Reduction for finding the maximum element, using a comparator.  
 The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.29 `__gnu_cxx::__detail::__mini_vector<_Tp>` Class Template Reference

```
#include <bitmap_allocator.h>
```

### Public Types

- typedef const `_Tp` & `const_reference`
- typedef std::ptrdiff\_t `difference_type`
- typedef pointer `iterator`
- typedef `_Tp *` `pointer`
- typedef `_Tp &` `reference`

- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator \_\_pos) throw ()
- void **insert** (iterator \_\_pos, const\_reference \_\_x)
- reference **operator[]** (const size\_type \_\_pos) const throw ()
- void **pop\_back** () throw ()
- void **push\_back** (const\_reference \_\_x)
- size\_type **size** () const throw ()

#### 5.29.1 Detailed Description

template<typename \_Tp>  
class \_\_gnu\_cxx::\_\_detail::\_\_mini\_vector< \_Tp >

\_\_mini\_vector<> is a stripped down version of the full-fledged std::vector<>. It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present.
2. Used ONLY for PODs.
3. No Allocator template argument. Uses operator new() to get memory, and operator delete() to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

#### 5.30 \_\_gnu\_parallel::\_\_mismatch\_selector Struct Reference

```
#include <find_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_mismatch\_selector:



## Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

### 5.30.1 Detailed Description

Test inverted predicate on a single element.

### 5.30.2 Member Function Documentation

#### `_M_sequential_algorithm()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__mismatch_selector::_M_sequential_algorithm (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

#### `operator()()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__mismatch_selector::operator() (
    _RAIter1 __i1,
    _RAIter2 __i2,
    _Pred __pred) [inline]
```

Test on one position.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

### 5.31 `__gnu_cxx::__mt_alloc<_Tp, _Poolp>` Class Template Reference

```
#include <mt_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`:



#### Public Types

- typedef `_Poolp` **\_\_policy\_type**
- typedef `_Poolp::pool_type` **\_\_pool\_type**
- typedef `const _Tp *` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Tp *` **pointer**
- typedef `std::true_type` **propagate\_on\_container\_move\_assignment**
- typedef `_Tp &` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- `__mt_alloc` (`const __mt_alloc &`) `noexcept`
- `template<typename _Tp1, typename _Poolp1>`  
`__mt_alloc` (`const __mt_alloc<_Tp1, _Poolp1> &`) `noexcept`
- `const __pool_base::_Tune` **M\_get\_options** ()
- `void` **M\_set\_options** (`__pool_base::_Tune` `__t`)
- `const_pointer` **address** (`const_reference` `__x`) `const noexcept`
- `pointer` **address** (`reference` `__x`) `const noexcept`
- `pointer` **allocate** (`size_type` `__n`, `const void *` `__p`)
- `template<typename _Up, typename... _Args>`  
`void` **construct** (`_Up *` `__p`, `_Args &&...` `__args`)
- `void` **deallocate** (`pointer` `__p`, `size_type` `__n`)
- `template<typename _Up>`  
`void` **destroy** (`_Up *` `__p`)
- `size_type` **max\_size** () `const noexcept`

### 5.31.1 Detailed Description

```
template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >>
class __gnu_cxx::__mt_alloc< _Tp, _Poolp >
```

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt\\_allocator.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt_allocator.html)

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.32 \_\_gnu\_cxx::\_\_mt\_alloc\_base< \_Tp > Class Template Reference

```
#include <mt_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::\_\_mt\_alloc\_base< \_Tp >:



### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **address** (reference \_\_x) const noexcept
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- template<typename \_Up>  
void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept



### 5.32.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__mt_alloc_base< _Tp >
```

Base class for `_Tp` dependent member functions.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

### 5.33 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIter3, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

```
#include <multiway_merge.h>
```

#### Public Member Functions

- `_RAIter3 operator()` (`_RAIter3 __seqs_begin`, `_RAIter3 __seqs_end`, `_RAIter3 __target`, `__↵`  
`DifferenceTp __length`, `_Compare __comp`)

#### 5.33.1 Detailed Description

```
template<bool __sentinels, typename _RAIter3, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIter3, _RAIter3,
_DifferenceTp, _Compare >
```

Switch for 3-way merging with `__sentinels` turned off.

Note that 3-way merging is always stable!

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 5.34 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIter3, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

```
#include <multiway_merge.h>
```

#### Public Member Functions

- `_RAIter3 operator()` (`_RAIter3 __seqs_begin`, `_RAIter3 __seqs_end`, `_RAIter3 __target`, `__↵`  
`DifferenceTp __length`, `_Compare __comp`)
- `_RAIter3 operator()` (`_RAIter3 __seqs_begin`, `_RAIter3 __seqs_end`, `_RAIter3 __target`, `__↵`  
`DifferenceTp __length`, `_Compare __comp`)

#### 5.34.1 Detailed Description

```
template<typename _RAIter3, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIter3, _RAIter3, __↵
_DifferenceTp, _Compare >
```

Switch for 3-way merging with `__sentinels` turned on.

Note that 3-way merging is always stable!

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 5.35 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

```
#include <multiway_merge.h>
```

#### Public Member Functions

- `_RAIter3 operator() ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _↵ DifferenceTp __length, _Compare __comp)`

#### 5.35.1 Detailed Description

```
template<bool __sentinels, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
```

```
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned off.

Note that 4-way merging is always stable!

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 5.36 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

```
#include <multiway_merge.h>
```

#### Public Member Functions

- `_RAIter3 operator() ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _↵ DifferenceTp __length, _Compare __comp)`
- `_RAIter3 operator() ( _RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _↵ DifferenceTp __length, _Compare __comp)`

#### 5.36.1 Detailed Description

```
template<typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
```

```
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _↵ DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned on.

Note that 4-way merging is always stable!

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 5.37 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

```
#include <multiway_merge.h>
```

## Public Member Functions

- **\_RAIter3 operator()** (\_RAIterliterator \_\_seqs\_begin, \_RAIterliterator \_\_seqs\_end, \_RAIter3 \_\_target, const typename [std::iterator\\_traits](#)< typename [std::iterator\\_traits](#)< \_RAIterliterator >::value\_type::first\_type >::value\_type & \_\_sentinel, \_DifferenceTp \_\_length, \_Compare \_\_comp)

### 5.37.1 Detailed Description

```
template<bool __sentinels, bool __stable, typename _RAIterliterator, typename _RAIter3, typename _↵
_DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterliterator,
_RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with \_\_sentinels turned on.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 5.38 \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel\_switch< false, \_\_stable, \_RAIterliterator, \_RAIter3, \_DifferenceTp, \_Compare > Struct Template Reference

```
#include <multiway_merge.h>
```

## Public Member Functions

- **\_RAIter3 operator()** (\_RAIterliterator \_\_seqs\_begin, \_RAIterliterator \_\_seqs\_end, \_RAIter3 \_\_target, const typename [std::iterator\\_traits](#)< typename [std::iterator\\_traits](#)< \_RAIterliterator >::value\_type::first\_type >::value\_type & \_\_sentinel, \_DifferenceTp \_\_length, \_Compare \_\_comp)
- **\_RAIter3 operator()** (\_RAIterliterator \_\_seqs\_begin, \_RAIterliterator \_\_seqs\_end, \_RAIter3 \_\_target, const typename [std::iterator\\_traits](#)< typename [std::iterator\\_traits](#)< \_RAIterliterator >::value\_type::first\_type >::value\_type & \_\_sentinel, \_DifferenceTp \_\_length, \_Compare \_\_comp)

### 5.38.1 Detailed Description

```
template<bool __stable, typename _RAIterliterator, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterliterator, _↵
_RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with \_\_sentinels turned off.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 5.39 std::\_\_new\_allocator< \_Tp > Class Template Reference

```
#include <memory>
```

Inheritance diagram for std::\_\_new\_allocator< \_Tp >:



### Public Types

- typedef std::ptrdiff\_t **difference\_type**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **\_\_new\_allocator** (const [\\_\\_new\\_allocator](#) &) noexcept
- template<typename \_Tp1>  
constexpr **\_\_new\_allocator** (const [\\_\\_new\\_allocator](#)< \_Tp1 > &) noexcept
- \_Tp \* **allocate** (size\_type \_\_n, const void \* =static\_cast< const void \* >(0))
- void **deallocate** (\_Tp \* \_\_p, size\_type \_\_n)
- [\\_\\_new\\_allocator](#) & **operator=** (const [\\_\\_new\\_allocator](#) &) = default

### Friends

- template<typename \_Up>  
constexpr bool **operator==** (const [\\_\\_new\\_allocator](#) &, const [\\_\\_new\\_allocator](#)< \_Up > &) noexcept

#### 5.39.1 Detailed Description

```
template<typename _Tp>
class std::__new_allocator< _Tp >
```

An allocator that uses global `new`, as per C++03 [20.4.1].  
This is precisely the allocator defined in the C++ Standard.

- all allocation calls `operator new`

- all deallocation calls `operator delete`

This is the default base-class implementation of `std::allocator`, and is also the base-class of the `__gnu_cxx::new_allocator` extension. You should use either `std::allocator` or `__gnu_cxx::new_allocator` instead of using this directly.

#### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

The documentation for this class was generated from the following file:

- [bits/new\\_allocator.h](#)

### 5.40 `std::__numeric_limits_base` Struct Reference

```
#include <limits>
```

Inheritance diagram for std::\_\_numeric\_limits\_base:



#### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int [digits10](#)

- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)
- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)
- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

#### 5.40.1 Detailed Description

Part of `std::numeric_limits`.

The `static const` members are usable as integral constant expressions.

##### Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the `std::numeric_limits` class.

#### 5.40.2 Member Data Documentation

##### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Referenced by [std::generate\\_canonical\(\)](#), [std::independent\\_bits\\_engine<\\_RandomNumberEngine, \\_\\_w, \\_UIntType >::operator\(\)\(\)](#), [std::operator<<\(\)](#), and [std::linear\\_congruential\\_engine< uint\\_fast32\\_t, 16807UL, 0UL, 2147483647UL >::seed\(\)](#).

##### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

##### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

##### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving `+`, `-`, or `*` on values of that type whose result would fall outside the range `[min(),max()]`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.



**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.41 `__gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread >` Struct Template Reference

```
#include <mt_allocator.h>
```

### 5.41.1 Detailed Description

```
template<typename _Tp, template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >
```

Policy for individual `__pool` objects.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.42 `__gnu_cxx::__pool<_Thread>` Class Template Reference

### 5.42.1 Detailed Description

```
template<bool _Thread>
class __gnu_cxx::__pool<_Thread>
```

Data describing the underlying memory pool, parameterized on threading support.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.43 `__gnu_cxx::__pool<false>` Class Reference

```
#include <mt_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__pool<false>`:



### Public Types

- typedef unsigned short int **\_Binmap\_type**
- typedef std::size\_t **size\_t**

### Public Member Functions

- **\_\_pool** (const `__pool_base::__Tune` & `__tune`)
- void **\_M\_adjust\_freelist** (const `_Bin_record` &, `_Block_record` \*, `size_t`)
- bool **\_M\_check\_threshold** (`size_t` `__bytes`)
- void **\_M\_destroy** () throw ()
- `size_t` **\_M\_get\_align** ()
- const `_Bin_record` & **\_M\_get\_bin** (`size_t` `__which`)

- `size_t _M_get_binmap (size_t __bytes)`
- `const _Tune & _M_get_options () const`
- `size_t _M_get_thread_id ()`
- `void _M_initialize_once ()`
- `void _M_reclaim_block (char *__p, size_t __bytes) throw ()`
- `char * _M_reserve_block (size_t __bytes, const size_t __thread_id)`
- `void _M_set_options (_Tune __t)`

### Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

#### 5.43.1 Detailed Description

Specialization for single thread.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

#### 5.44 `__gnu_cxx::__pool< true >` Class Reference

```
#include <mt_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__pool< true >`:



### Public Types

- `typedef unsigned short int _Binmap_type`
- `typedef std::size_t size_t`

### Public Member Functions

- `__pool (const __pool_base::_Tune &__tune)`
- `void _M_adjust_freelist (const _Bin_record &__bin, _Block_record * __block, size_t __thread_id)`
- `bool _M_check_threshold (size_t __bytes)`
- `void _M_destroy () throw ()`

- void **\_M\_destroy\_thread\_key** (void \*) throw ()
- size\_t **\_M\_get\_align** ()
- const \_Bin\_record & **\_M\_get\_bin** (size\_t \_\_which)
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- size\_t **\_M\_get\_thread\_id** ()
- void **\_M\_initialize** (\_\_destroy\_handler)
- void **\_M\_initialize\_once** ()
- void **\_M\_reclaim\_block** (char \*\_\_p, size\_t \_\_bytes) throw ()
- char \* **\_M\_reserve\_block** (size\_t \_\_bytes, const size\_t \_\_thread\_id)
- void **\_M\_set\_options** (\_Tune \_\_t)

#### Protected Attributes

- \_Binmap\_type \* **\_M\_binmap**
- bool **\_M\_init**
- \_Tune **\_M\_options**

#### 5.44.1 Detailed Description

Specialization for thread enabled, via gthreads.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

#### 5.45 \_\_gnu\_cxx::\_\_pool\_alloc<\_Tp> Class Template Reference

```
#include <pool_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::\_\_pool\_alloc<\_Tp>:



#### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**

- typedef `_Tp` & **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- `__pool_alloc` (const `__pool_alloc` &) noexcept
- template<typename `_Tp1`>  
  `__pool_alloc` (const `__pool_alloc`< `_Tp1` > &) noexcept
- const\_pointer **address** (const\_reference `__x`) const noexcept
- pointer **address** (reference `__x`) const noexcept
- pointer **allocate** (size\_type `__n`, const void \*=0)
- template<typename `_Up`, typename... `_Args`>  
  void **construct** (`_Up` \*`__p`, `_Args` &&... `__args`)
- void **deallocate** (pointer `__p`, size\_type `__n`)
- template<typename `_Up`>  
  void **destroy** (`_Up` \*`__p`)
- size\_type **max\_size** () const noexcept

#### 5.45.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__pool_alloc< _Tp >
```

Allocator using a memory pool with a single lock.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

#### 5.46 \_\_gnu\_cxx::\_\_pool\_alloc\_base Class Reference

```
#include <pool_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__pool_alloc_base`:



### Protected Types

- enum { **\_S\_free\_list\_size** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_align** }

**Protected Member Functions**

- `char * M_allocate_chunk (size_t __n, int &__nobjs)`
- `_Obj *volatile * M_get_free_list (size_t __bytes) throw ()`
- `__mutex & M_get_mutex () throw ()`
- `void * M_refill (size_t __n)`
- `size_t M_round_up (size_t __bytes)`

**Static Protected Attributes**

- `static char * S_end_free`
- `static _Obj *volatile S_free_list [S_free_list_size]`
- `static size_t S_heap_size`
- `static char * S_start_free`

**5.46.1 Detailed Description**

Base class for `__pool_alloc`.

Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties:

1. If globally mandated, then allocate objects from `new`
2. If the clients request an object of size `> _S_max_bytes`, the resulting object will be obtained directly from `new`
3. In all other cases, we allocate an object of size exactly `_S_round_up(requested_size)`. Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

**5.47 `__gnu_cxx::__pool_base` Struct Reference**

```
#include <mt_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__pool_base`:



### Public Types

- typedef unsigned short int **\_Binmap\_type**
- typedef std::size\_t **size\_t**

### Public Member Functions

- **\_\_pool\_base** (const \_Tune &\_\_options)
- bool **\_M\_check\_threshold** (size\_t \_\_bytes)
- size\_t **\_M\_get\_align** ()
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- void **\_M\_set\_options** (\_Tune \_\_t)

### Protected Attributes

- \_Binmap\_type \* **\_M\_binmap**
- bool **\_M\_init**
- \_Tune **\_M\_options**

#### 5.47.1 Detailed Description

Base class for pool object.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

#### 5.48 `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>` Class Template Reference

```
#include <rc_string_base.h>
```

Inheritance diagram for `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>`:



### Public Types

- `typedef _Util_Base::_CharT_alloc_type _CharT_alloc_type`
- `typedef __vstring_utility<_CharT, _Traits, _Alloc> _Util_Base`
- `typedef _Alloc allocator_type`
- `typedef _CharT_alloc_type::size_type size_type`
- `typedef _Traits traits_type`
- `typedef _Traits::char_type value_type`

### Public Member Functions

- `_rc_string_base (\_rc\_string\_base &&__rcs)`
- `template<typename _InputIterator>  
_rc_string_base (_InputIterator __beg, _InputIterator __end, const _Alloc &__a)`
- `_rc_string_base (const \_rc\_string\_base &__rcs)`



- `__rc_string_base` (const `_Alloc` & `__a`)
- `__rc_string_base` (size\_type `__n`, `_CharT` `__c`, const `_Alloc` & `__a`)
- void `_M_assign` (const `__rc_string_base` & `__rcs`)
- size\_type `_M_capacity` () const
- void `_M_clear` ()
- bool `_M_compare` (const `__rc_string_base` &) const
- bool `_M_compare` (const `__rc_string_base` & `__rcs`) const
- bool `_M_compare` (const `__rc_string_base` & `__rcs`) const
- `_CharT` \* `_M_data` () const
- void `_M_erase` (size\_type `__pos`, size\_type `__n`)
- allocator\_type & `_M_get_allocator` ()
- const allocator\_type & `_M_get_allocator` () const
- bool `_M_is_shared` () const
- void `_M_leak` ()
- size\_type `_M_length` () const
- size\_type `_M_max_size` () const
- void `_M_mutate` (size\_type `__pos`, size\_type `__len1`, const `_CharT` \* `__s`, size\_type `__len2`)
- void `_M_reserve` (size\_type `__res`)
- void `_M_set_leaked` ()
- void `_M_set_length` (size\_type `__n`)
- void `_M_swap` (`__rc_string_base` & `__rcs`)
- template<typename `_InIterator`>  
`_CharT` \* `_S_construct` (`_InIterator` `__beg`, `_InIterator` `__end`, const `_Alloc` & `__a`, `std::forward_iterator_tag`)

### Protected Types

- typedef `__gnu_cxx::__normal_iterator`< const\_pointer, `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `__rc_string_base` > > `__const_rc_iterator`
- typedef `__gnu_cxx::__normal_iterator`< const\_pointer, `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `__sso_string_base` > > `__const_sso_iterator`
- typedef `__gnu_cxx::__normal_iterator`< pointer, `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `__rc_string_base` > > `__rc_iterator`
- typedef `__gnu_cxx::__normal_iterator`< pointer, `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `__sso_string_base` > > `__sso_iterator`
- typedef `__alloc_traits`< `_CharT`, `_Alloc` > `_CharT_alloc_traits`
- typedef `_CharT_alloc_traits::const_pointer` `const_pointer`
- typedef `_CharT_alloc_traits::difference_type` `difference_type`
- typedef `_CharT_alloc_traits::pointer` `pointer`

### Static Protected Member Functions

- static void `_S_assign` (`_CharT` \* `__d`, size\_type `__n`, `_CharT` `__c`)
- static int `_S_compare` (size\_type `__n1`, size\_type `__n2`)
- static void `_S_copy` (`_CharT` \* `__d`, const `_CharT` \* `__s`, size\_type `__n`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `__const_rc_iterator` `__k1`, `__const_rc_iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `__const_sso_iterator` `__k1`, `__const_sso_iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `__rc_iterator` `__k1`, `__rc_iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `__sso_iterator` `__k1`, `__sso_iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `_CharT` \* `__k1`, `_CharT` \* `__k2`)
- template<typename `_Iterator`>  
static void `_S_copy_chars` (`_CharT` \* `__p`, `_Iterator` `__k1`, `_Iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, const `_CharT` \* `__k1`, const `_CharT` \* `__k2`)
- static void `_S_move` (`_CharT` \* `__d`, const `_CharT` \* `__s`, size\_type `__n`)

## 5.48.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >
```

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).  
A string looks like this:

```

                                     [_Rep]
                                     _M_length
[_rc_string_base<char_type>]         _M_capacity
_M_dataplus                         _M_refcount
_M_p ----->                       unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

The documentation for this class was generated from the following file:

- [rc\\_string\\_base.h](#)

5.49 `std::tr2::__reflection_typelist<_Elements>` Struct Template Reference

## 5.49.1 Detailed Description

```
template<typename... _Elements>
struct std::tr2::__reflection_typelist<_Elements>
```

See N2965: Type traits and base classes by Michael Spertus Simple typelist. Compile-time list of types.  
The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

5.50 `std::tr2::__reflection_typelist<_First, _Rest...>` Struct Template Reference

```
#include <type_traits>
```

## Public Types

- typedef [std::false\\_type](#) `empty`

## 5.50.1 Detailed Description

```
template<typename _First, typename... _Rest>
struct std::tr2::__reflection_typelist<_First, _Rest...>
```

Partial specialization.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 5.51 `std::tr2::__reflection_typelist<>` Struct Reference

```
#include <type_traits>
```

### Public Types

- typedef `std::true_type` `empty`

#### 5.51.1 Detailed Description

Specialization for an empty typelist.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 5.52 `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`:



### Public Member Functions

- `__replace_if_selector` (`const _Tp &__new_val`)
- `bool operator()` (`_Op &__o, _It __i`)

### Public Attributes

- `const _Tp &__new_val`
- `_It __M_finish_iterator`

#### 5.52.1 Detailed Description

```
template<typename _It, typename _Op, typename _Tp>
struct __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>
```

`std::replace()` selector.

### 5.52.2 Constructor & Destructor Documentation

#### `__replace_if_selector()`

```
template<typename _It, typename _Op, typename _Tp>
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__replace_if_selector (
    const _Tp & __new_val) [inline], [explicit]
```

Constructor.

#### Parameters

|                        |                        |
|------------------------|------------------------|
| <code>__new_val</code> | Value to replace with. |
|------------------------|------------------------|

References [\\_\\_new\\_val](#).

### 5.52.3 Member Function Documentation

#### `operator()()`

```
template<typename _It, typename _Op, typename _Tp>
bool __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator() (
    _Op & __o,
    _It __i) [inline]
```

Functor execution.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator.                    |
| <code>__i</code> | iterator referencing object. |

References [\\_\\_new\\_val](#).

### 5.52.4 Member Data Documentation

#### `__new_val`

```
template<typename _It, typename _Op, typename _Tp>
const _Tp& __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val
```

Value to replace with.

Referenced by [\\_\\_replace\\_if\\_selector\(\)](#), and [operator\(\)\(\)](#).

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.53 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__replace_selector<_It,_Tp>`:



### Public Member Functions

- `__replace_selector` (const `_Tp` & `__new_val`)
- bool `operator()` (`_Tp` & `__v`, `_It` `__i`)

### Public Attributes

- const `_Tp` & `__new_val`
- `_It` `__M_finish_iterator`

### 5.53.1 Detailed Description

```
template<typename _It, typename _Tp>
struct __gnu_parallel::__replace_selector<_It,_Tp>
```

`std::replace()` selector.

### 5.53.2 Constructor & Destructor Documentation

#### `__replace_selector()`

```
template<typename _It, typename _Tp>
__gnu_parallel::__replace_selector<_It,_Tp>::__replace_selector (
    const _Tp & __new_val) [inline], [explicit]
```

Constructor.

#### Parameters

|                        |                        |
|------------------------|------------------------|
| <code>__new_val</code> | Value to replace with. |
|------------------------|------------------------|

References `__new_val`.

### 5.53.3 Member Function Documentation

#### `operator()()`

```
template<typename _It, typename _Tp>
bool __gnu_parallel::__replace_selector< _It, _Tp >::operator() (
    _Tp & __v,
    _It __i) [inline]
```

Functor execution.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__v</code> | Current value.               |
| <code>__i</code> | iterator referencing object. |

References [\\_\\_new\\_val](#).

### 5.53.4 Member Data Documentation

#### `__new_val`

```
template<typename _It, typename _Tp>
const _Tp& __gnu_parallel::__replace_selector< _It, _Tp >::__new_val
```

Value to replace with.

Referenced by [\\_\\_replace\\_selector\(\)](#), and [operator\(\)\(\)](#).

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.54 `__gnu_cxx::__scoped_lock` Class Reference

```
#include <concurrency.h>
```

### Public Types

- typedef `__mutex` `__mutex_type`

### Public Member Functions

- `__scoped_lock` (`__mutex_type` & `__name`)

#### 5.54.1 Detailed Description

Scoped lock idiom.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

## 5.55 \_\_gnu\_parallel::\_\_transform1\_selector<\_It> Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_transform1\_selector<\_It>:



### Public Member Functions

- template<typename \_Op>  
bool [operator\(\)](#) (\_Op &\_\_o, \_It \_\_i)

### Public Attributes

- [\\_It \\_M\\_finish\\_iterator](#)

#### 5.55.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__transform1_selector<_It>
```

std::transform() \_\_selector, one input sequence variant.

#### 5.55.2 Member Function Documentation

##### operator>()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__transform1_selector<_It>::operator() (
    _Op & __o,
    _It __i) [inline]
```

Functor execution.

##### Parameters

|                     |                              |
|---------------------|------------------------------|
| <a href="#">__o</a> | Operator.                    |
| <a href="#">__i</a> | iterator referencing object. |

### 5.55.3 Member Data Documentation

#### `_M_finish_iterator`

```
template<typename _It>
```

```
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]  
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.56 \_\_gnu\_parallel::\_\_transform2\_selector<\_It> Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__transform2_selector<_It>`:



### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

### 5.56.1 Detailed Description

```
template<typename _It>
```

```
struct __gnu_parallel::__transform2_selector< _It >
```

`std::transform()` \_\_selector, two input sequences variant.

### 5.56.2 Member Function Documentation

#### `operator()()`

```
template<typename _It>
```

```
template<typename _Op>
```

```
bool __gnu_parallel::__transform2_selector< _It >::operator() (
```



```

    _Op & __o,
    _It __i) [inline]

```

Functor execution.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator.                    |
| <code>__i</code> | iterator referencing object. |

### 5.56.3 Member Data Documentation

#### `_M_finish_iterator`

```

template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).

```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.57 `__gnu_parallel::__unary_negate< _Predicate, argument_type >` Class Template Reference

```

#include <base.h>

```

Inheritance diagram for `__gnu_parallel::__unary_negate< _Predicate, argument_type >`:



#### Public Types

- typedef `argument_type` `argument_type`
- typedef `bool` `result_type`

### Public Member Functions

- `__unary_negate` (const `_Predicate` &\_\_x)
- `bool operator()` (const `argument_type` &\_\_x)

### Protected Attributes

- `_Predicate _M_pred`

#### 5.57.1 Detailed Description

```
template<typename _Predicate, typename argument\_type>
class __gnu_parallel::__unary_negate<_Predicate, argument\_type>
```

Similar to `std::unary_negate`, but giving the argument types explicitly.

#### 5.57.2 Member Typedef Documentation

##### `argument_type`

```
typedef argument\_type std::unary_function< argument\_type, bool >::argument_type [inherited]
argument\_type is the type of the argument
```

##### `result_type`

```
typedef bool std::unary_function< argument\_type, bool >::result_type [inherited]
result\_type is the return type
```

The documentation for this class was generated from the following file:

- [base.h](#)

## 5.58 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference

```
#include <vstring.h>
```

### Public Types

- typedef `_Alloc` `allocator_type`
- typedef `__gnu_cxx::__normal_iterator< const_pointer, \_\_versa\_string>` `const_iterator`
- typedef `_CharT_alloc_traits::const_pointer` `const_pointer`
- typedef `const value_type &` `const_reference`
- typedef `std::reverse\_iterator< const\_iterator >` `const_reverse_iterator`
- typedef `_CharT_alloc_type::difference_type` `difference_type`
- typedef `__gnu_cxx::__normal_iterator< pointer, \_\_versa\_string>` `iterator`
- typedef `_CharT_alloc_traits::pointer` `pointer`
- typedef `value_type &` `reference`
- typedef `std::reverse\_iterator< iterator >` `reverse_iterator`
- typedef `_CharT_alloc_type::size_type` `size_type`
- typedef `_Traits` `traits_type`
- typedef `_Traits::char_type` `value_type`

## Public Member Functions

- [\\_\\_versa\\_string](#) ([\\_\\_versa\\_string](#) &&\_\_str) noexcept
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
[\\_\\_versa\\_string](#) (\_InputIterator \_\_beg, \_InputIterator \_\_end, const \_Alloc &\_\_a= \_Alloc())
- [\\_\\_versa\\_string](#) (const [\\_\\_versa\\_string](#) &\_\_str)
- [\\_\\_versa\\_string](#) (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n, const \_Alloc &\_\_a)
- [\\_\\_versa\\_string](#) (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[npos](#))
- [\\_\\_versa\\_string](#) (const \_Alloc &\_\_a= \_Alloc()) noexcept
- [\\_\\_versa\\_string](#) (const \_CharT \*\_\_s, const \_Alloc &\_\_a= \_Alloc())
- [\\_\\_versa\\_string](#) (const \_CharT \*\_\_s, size\_type \_\_n, const \_Alloc &\_\_a= \_Alloc())
- [\\_\\_versa\\_string](#) (size\_type \_\_n, \_CharT \_\_c, const \_Alloc &\_\_a= \_Alloc())
- [\\_\\_versa\\_string](#) (std::initializer\_list<\_CharT > \_\_l, const \_Alloc &\_\_a= \_Alloc())
- ~[\\_\\_versa\\_string](#) () noexcept
- template<typename \_InputIterator>  
[\\_\\_versa\\_string](#)<\_CharT, \_Traits, \_Alloc, \_Base > & **[M\\_replace\\_dispatch](#)** (const\_iterator \_\_i1, const\_iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2, std::false\_type)
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
[\\_\\_versa\\_string](#) & **[append](#)** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [\\_\\_versa\\_string](#) & **[append](#)** (const [\\_\\_versa\\_string](#) &\_\_str)
- [\\_\\_versa\\_string](#) & **[append](#)** (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [\\_\\_versa\\_string](#) & **[append](#)** (const \_CharT \*\_\_s)
- [\\_\\_versa\\_string](#) & **[append](#)** (const \_CharT \*\_\_s, size\_type \_\_n)
- [\\_\\_versa\\_string](#) & **[append](#)** (size\_type \_\_n, \_CharT \_\_c)
- [\\_\\_versa\\_string](#) & **[append](#)** (std::initializer\_list<\_CharT > \_\_l)
- [\\_\\_versa\\_string](#) & **[assign](#)** ([\\_\\_versa\\_string](#) &&\_\_str) noexcept
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
[\\_\\_versa\\_string](#) & **[assign](#)** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [\\_\\_versa\\_string](#) & **[assign](#)** (const [\\_\\_versa\\_string](#) &\_\_str)
- [\\_\\_versa\\_string](#) & **[assign](#)** (const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [\\_\\_versa\\_string](#) & **[assign](#)** (const \_CharT \*\_\_s)
- [\\_\\_versa\\_string](#) & **[assign](#)** (const \_CharT \*\_\_s, size\_type \_\_n)
- [\\_\\_versa\\_string](#) & **[assign](#)** (size\_type \_\_n, \_CharT \_\_c)
- [\\_\\_versa\\_string](#) & **[assign](#)** (std::initializer\_list<\_CharT > \_\_l)
- reference **[at](#)** (size\_type \_\_n)
- const\_reference **[at](#)** (size\_type \_\_n) const
- const\_reference **[back](#)** () const noexcept
- reference **[back](#)** () noexcept
- const\_iterator **[begin](#)** () const noexcept
- iterator **[begin](#)** () noexcept
- const \_CharT \* **[c\\_str](#)** () const noexcept
- size\_type **[capacity](#)** () const noexcept
- const\_iterator **[cbegin](#)** () const noexcept
- const\_iterator **[cend](#)** () const noexcept
- void **[clear](#)** () noexcept
- int **[compare](#)** (const [\\_\\_versa\\_string](#) &\_\_str) const
- int **[compare](#)** (const \_CharT \*\_\_s) const
- int **[compare](#)** (size\_type \_\_pos, size\_type \_\_n, const [\\_\\_versa\\_string](#) &\_\_str) const
- int **[compare](#)** (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s) const
- int **[compare](#)** (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2) const
- int **[compare](#)** (size\_type \_\_pos1, size\_type \_\_n1, const [\\_\\_versa\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n2) const

- `size_type copy` (`_CharT * __s`, `size_type __n`, `size_type __pos=0`) `const`
- `const_reverse_iterator crbegin` () `const noexcept`
- `const_reverse_iterator crend` () `const noexcept`
- `const _CharT * data` () `const noexcept`
- `bool empty` () `const noexcept`
- `const_iterator end` () `const noexcept`
- `iterator end` () `noexcept`
- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator erase` (`const_iterator __position`)
- `__versa_string & erase` (`size_type __pos=0`, `size_type __n=npos`)
- `size_type find` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find` (`const __versa_string & __str`, `size_type __pos=0`) `const noexcept`
- `size_type find` (`const _CharT * __s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find` (`const _CharT * __s`, `size_type __pos=0`) `const`
- `size_type find_first_not_of` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_not_of` (`const __versa_string & __str`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_not_of` (`const _CharT * __s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_first_not_of` (`const _CharT * __s`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_of` (`const __versa_string & __str`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_of` (`const _CharT * __s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_first_of` (`const _CharT * __s`, `size_type __pos=0`) `const`
- `size_type find_last_not_of` (`_CharT __c`, `size_type __pos=npo`) `const noexcept`
- `size_type find_last_not_of` (`const __versa_string & __str`, `size_type __pos=npo`) `const noexcept`
- `size_type find_last_not_of` (`const _CharT * __s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_not_of` (`const _CharT * __s`, `size_type __pos=npo`) `const`
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=npo`) `const noexcept`
- `size_type find_last_of` (`const __versa_string & __str`, `size_type __pos=npo`) `const noexcept`
- `size_type find_last_of` (`const _CharT * __s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_of` (`const _CharT * __s`, `size_type __pos=npo`) `const`
- `const_reference front` () `const noexcept`
- `reference front` () `noexcept`
- `allocator_type get_allocator` () `const noexcept`
- `iterator insert` (`const_iterator __p`, `_CharT __c`)
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`iterator insert` (`const_iterator __p`, `_InputIterator __beg`, `_InputIterator __end`)
- `iterator insert` (`const_iterator __p`, `size_type __n`, `_CharT __c`)
- `iterator insert` (`const_iterator __p`, `std::initializer_list<_CharT> __l`)
- `__versa_string & insert` (`size_type __pos`, `const _CharT * __s`)
- `__versa_string & insert` (`size_type __pos`, `const _CharT * __s`, `size_type __n`)
- `__versa_string & insert` (`size_type __pos`, `size_type __n`, `_CharT __c`)
- `__versa_string & insert` (`size_type __pos1`, `const __versa_string & __str`)
- `__versa_string & insert` (`size_type __pos1`, `const __versa_string & __str`, `size_type __pos2`, `size_type __n`)
- `size_type length` () `const noexcept`
- `size_type max_size` () `const noexcept`
- `__versa_string & operator+=` (`_CharT __c`)
- `__versa_string & operator+=` (`const __versa_string & __str`)
- `__versa_string & operator+=` (`const _CharT * __s`)
- `__versa_string & operator+=` (`std::initializer_list<_CharT> __l`)
- `__versa_string & operator=` (`__versa_string && __str`) `noexcept`
- `__versa_string & operator=` (`_CharT __c`)

- `__versa_string & operator=` (const `__versa_string` & \_\_str)
- `__versa_string & operator=` (const `_CharT *` \_\_s)
- `__versa_string & operator=` (std::initializer\_list< `_CharT` > \_\_l)
- const\_reference `operator[]` (size\_type \_\_pos) const noexcept
- reference `operator[]` (size\_type \_\_pos) noexcept
- void `pop_back` ()
- void `push_back` (`_CharT` \_\_c)
- `const_reverse_iterator rbegin` () const noexcept
- `reverse_iterator rbegin` () noexcept
- `const_reverse_iterator rend` () const noexcept
- `reverse_iterator rend` () noexcept
- `__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, `_CharT *` \_\_k1, `_CharT *` \_\_k2)
- template<class `_InputIterator`, typename = std::\_RequireInputIter< `_InputIterator` >>  
`__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, `_InputIterator` \_\_k1, `_InputIterator` \_\_k2)
- `__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, const `__versa_string` & \_\_str)
- `__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, const `_CharT *` \_\_k1, const `_CharT *` \_\_k2)
- `__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, const `_CharT *` \_\_s)
- `__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, const `_CharT *` \_\_s, size\_type \_\_n)
- `__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- `__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- `__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, size\_type \_\_n, `_CharT` \_\_c)
- `__versa_string & replace` (const\_iterator \_\_i1, const\_iterator \_\_i2, std::initializer\_list< `_CharT` > \_\_l)
- `__versa_string & replace` (size\_type \_\_pos, size\_type \_\_n, const `__versa_string` & \_\_str)
- `__versa_string & replace` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT *` \_\_s)
- `__versa_string & replace` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT *` \_\_s, size\_type \_\_n2)
- `__versa_string & replace` (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, `_CharT` \_\_c)
- `__versa_string & replace` (size\_type \_\_pos1, size\_type \_\_n1, const `__versa_string` & \_\_str, size\_type \_\_pos2, size\_type \_\_n2)
- void `reserve` (size\_type \_\_res\_arg=0)
- void `resize` (size\_type \_\_n)
- void `resize` (size\_type \_\_n, `_CharT` \_\_c)
- size\_type `rfind` (`_CharT` \_\_c, size\_type \_\_pos=npos) const noexcept
- size\_type `rfind` (const `__versa_string` & \_\_str, size\_type \_\_pos=npos) const noexcept
- size\_type `rfind` (const `_CharT *` \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `rfind` (const `_CharT *` \_\_s, size\_type \_\_pos=npos) const
- void `shrink_to_fit` () noexcept
- size\_type `size` () const noexcept
- `__versa_string substr` (size\_type \_\_pos=0, size\_type \_\_n=npos) const
- void `swap` (`__versa_string` & \_\_s) noexcept

## Static Public Attributes

- static const size\_type `npos`

### 5.58.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base>
class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >
```

Template class `__versa_string`.

Data structure managing sequences of characters and character-like objects.

### 5.58.2 Constructor & Destructor Documentation

#### `__versa_string()` [1/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (
    const _Alloc & __a = _Alloc()) [inline], [explicit], [noexcept]
```

Construct an empty string using allocator *a*.

Referenced by `compare()`, `compare()`, and `__gnu_cxx::__versa_string<char>::substr()`.

#### `__versa_string()` [2/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (
    const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]
```

Construct string with copy of value of `__str`.

##### Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

#### `__versa_string()` [3/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (
    __versa_string<_CharT, _Traits, _Alloc, _Base> && __str) [inline], [noexcept]
```

String move constructor.

##### Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The newly-constructed string contains the exact contents of `__str`. The contents of `__str` are a valid, but unspecified string.

#### `__versa_string()` [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (
    std::initializer_list<_CharT> __l,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string from an initializer list.

##### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__l</code> | <code>std::initializer_list</code> of characters. |
| <code>__a</code> | Allocator to use (default is default allocator).  |

**\_\_versa\_string()** [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos,
    size_type __n = npos) [inline]
```

Construct string as copy of a substring.

**Parameters**

|                    |                                                   |
|--------------------|---------------------------------------------------|
| <code>__str</code> | Source string.                                    |
| <code>__pos</code> | Index of first character to copy from.            |
| <code>__n</code>   | Number of characters to copy (default remainder). |

**\_\_versa\_string()** [6/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos,
    size_type __n,
    const _Alloc & __a) [inline]
```

Construct string as copy of a substring.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |
| <code>__a</code>   | Allocator to use.                      |

**\_\_versa\_string()** [7/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const _CharT * __s,
    size_type __n,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string initialized by a character array.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source character array.                          |
| <code>__n</code> | Number of characters to copy.                    |
| <code>__a</code> | Allocator to use (default is default allocator). |

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

#### `__versa_string()` [8/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const _CharT * __s,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a C string.

##### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source C string.                                 |
| <code>__a</code> | Allocator to use (default is default allocator). |

#### `__versa_string()` [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    size_type __n,
    _CharT __c,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string as multiple characters.

##### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__n</code> | Number of characters.                            |
| <code>__c</code> | Character to use.                                |
| <code>__a</code> | Allocator to use (default is default allocator). |

#### `__versa_string()` [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    _InputIterator __beg,
    _InputIterator __end,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a range.



## Parameters

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__beg</code> | Start of range.                                  |
| <code>__end</code> | End of range.                                    |
| <code>__a</code>   | Allocator to use (default is default allocator). |

`~__versa_string()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::~__versa_string () [inline], [noexcept]
Destroy the string instance.
```

## 5.58.3 Member Function Documentation

`append()` [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
    _InputIterator __first,
    _InputIterator __last) [inline]
Append a range of characters.
```

## Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

## Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

`append()` [2/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
Append a string to this string.
```

## Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

## Returns

Reference to this string.

Referenced by `__gnu_cxx::__versa_string< char >::append()`, `std::getline()`, `__gnu_cxx::operator+()`, `__gnu_cxx::operator+()`, `__gnu_cxx::operator+()`, `__gnu_cxx::__versa_string< char >::operator+=()`, `__gnu_cxx::__versa_string< char >::operator+=()`, `__gnu_cxx::__versa_string< char >::operator+=()`, and `resize()`.

**append()** [3/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append (
    const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str,
    size_type __pos,
    size_type __n) [inline]
```

Append a substring.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__str</code> | The string to append.                                       |
| <code>__pos</code> | Index of the first character of <code>str</code> to append. |
| <code>__n</code>   | The number of characters to append.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

**append()** [4/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append (
    const _CharT * __s) [inline]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

**append()** [5/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append (
    const _CharT * __s,
    size_type __n) [inline]
```

Append a C substring.

**Parameters**

|                                                                                                                                                                                                                           |                                     |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| <a href="#"></a><br><a href="#"></a><br><code>_s</code> | The C string to append.             |
| <a href="#"></a><br><a href="#"></a><br><code>_n</code> | The number of characters to append. |

**Returns**

Reference to this string.

**append() [6/7]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
    size_type __n,
    _CharT __c) [inline]
```

Append multiple characters.

**Parameters**

|                                                                                                                                                                                                                             |                                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| <a href="#"></a><br><a href="#"></a><br><code>_n</code>   | The number of characters to append. |
| <a href="#"></a><br><a href="#"></a><br><code>_c</code> | The character to use.               |

**Returns**

Reference to this string.

Appends n copies of c to this string.

**append() [7/7]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
    std::initializer_list< _CharT > __l) [inline]
```

Append an initializer\_list of characters.

**Parameters**

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                               |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| <a href="#"></a><br><a href="#"></a><br><a href="#"></a><br><a href="#"></a><br><a href="#"></a><br><code>l</code> | The initializer_list of characters to append. |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|

**Returns**

Reference to this string.

**assign()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign (
    __versa_string<_CharT, _Traits, _Alloc, _Base> && __str) [inline], [noexcept]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

**assign()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Set value to a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Sets value of string to characters in the range `[first,last)`.

**assign()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign (
    const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Referenced by `__gnu_cxx::__versa_string<char>::assign()`, `__gnu_cxx::__versa_string<char>::operator=()`, `__gnu_cxx::__versa_string<char>::operator=()`, `__gnu_cxx::__versa_string<char>::operator=()`, and `__gnu_cxx::__versa_string<char>::operator=()`.

**assign()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos,
    size_type __n) [inline]
```

Set value to a substring of a string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                             |
|--------------------------------|---------------------------------------------|
| <code>std::out_of_range</code> | if <code>__pos</code> is not a valid index. |
|--------------------------------|---------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

**assign()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    const _CharT * __s) [inline]
```

Set value to contents of a C string.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | The C string to use. |
|------------------|----------------------|

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

**assign()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    const _CharT * __s,
    size_type __n) [inline]
```

Set value to a C substring.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | The C string to use.         |
| <code>__n</code> | Number of characters to use. |

## Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

**assign()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (
    size_type __n,
    _CharT __c) [inline]
```

Set value to multiple characters.

## Parameters

|                  |                                 |
|------------------|---------------------------------|
| <code>__n</code> | Length of the resulting string. |
| <code>__c</code> | The character to use.           |

## Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

**assign()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (
    std::initializer_list<_CharT > __l) [inline]
```

Set value to an `initializer_list` of characters.

## Parameters

|                |                                                            |
|----------------|------------------------------------------------------------|
| <code>l</code> | The <code>initializer_list</code> of characters to assign. |
|----------------|------------------------------------------------------------|

## Returns

Reference to this string.

**at()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (
    size_type __n) [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

**at()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (
    size_type __n) const [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**back()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back () const [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

**back()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
reference \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>::back\(\) [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the string.

**begin()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
const_iterator \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>::begin\(\) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**begin()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
iterator \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>::begin\(\) [inline], [noexcept]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::crend\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::rend\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::rend\(\)](#).

**c\_str()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
const _CharT * \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>::c\_str\(\) const [inline],
[noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

**capacity()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
size_type \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>::capacity\(\) const [inline],
[noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::push\\_back\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::shrink\\_to\\_fit\(\)](#).

**cbegin()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
const_iterator \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>::cbegin\(\) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**cend()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
const_iterator \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>::cend\(\) const [inline],
[noexcept]
```



```
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::clear () [inline], [noexcept]
```

```
int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (
    const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str) const [inline]
```

Referenced by `__gnu_cxx::operator<()`, `__gnu_cxx::operator<()`, `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator>()`, `__gnu_cxx::operator>()`, `__gnu_cxx::operator>()`, `__gnu_cxx::operator>=()`, `__gnu_cxx::operator>=()`, and `__gnu_cxx::operator>=()`.

```
int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (
    const _CharT * __s) const
```

References `std::min()`, and `size()`.

**compare()** [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
    size_type __pos,
    size_type __n,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) const
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

References `__versa_string()`, `data()`, `std::min()`, and `size()`.

**compare()** [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s) const
```

Compare substring to a C string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

References `std::min()`.

**compare()** [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2) const
```

Compare substring against a character array.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | character array to compare against.    |
| <code>__n2</code>  | Number of characters of s.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(), __s, rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `__s` must have at least `n2` characters, `\0` has no special meaning.

References [std::min\(\)](#).

**compare()** [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
    size_type __pos1,
    size_type __n1,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos2,
    size_type __n2) const
```

Compare substring to a substring.

**Parameters**

|                     |                                               |
|---------------------|-----------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.        |
| <code>__n1</code>   | Number of characters in substring.            |
| <code>__str</code>  | String to compare against.                    |
| <code>__pos2</code> | Index of first character of substring of str. |
| <code>__n2</code>   | Number of characters in substring of str.     |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(), str.substr(pos2, n2).data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

References `__versa_string()`, `data()`, and `std::min()`.

**copy()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, ↵
_Traits, _Alloc, _Base>::copy (
    _CharT * __s,
    size_type __n,
    size_type __pos = 0) const
```

Copy substring into C string.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

**Returns**

Number of characters actually copied

**Exceptions**

|                                |                                   |
|--------------------------------|-----------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> . |
|--------------------------------|-----------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

**crbegin()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::crbegin ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**crend()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

### data()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT * __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data () const [inline],
[noexcept]
```

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Referenced by [compare\(\)](#), [compare\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::compare\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_first\\_not\\_of\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_first\\_of\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::operator==\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::operator==\(\)](#).

### empty()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

### end() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

### end() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::cbegin\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::rbegin\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::rbegin\(\)](#).

### erase() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
    const_iterator __first,
    const_iterator __last) [inline]
```

Remove a range of characters.

#### Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

---

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
    const_iterator __position) [inline]
```

Remove one character.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

**Returns**

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
    size_type __pos = 0,
    size_type __n = npos) [inline]
```

Remove characters.

**Parameters**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |
|--------------------------------|---------------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by [std::getline\(\)](#).

**find()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find (
    _CharT __c,
    size_type __pos = 0) const [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const
```

Find position of a C substring.

## Parameters

|                    |                                                           |
|--------------------|-----------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                       |
| <code>__pos</code> | Index of character to search from.                        |
| <code>__n</code>   | Number of characters from <code>__s</code> to search for. |

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<](#)

**find()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
    const _CharT * __s,
    size_type __pos = 0) const [inline]
```

Find position of a C string.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find\_first\_not\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_not_of (
    _CharT __c,
    size_type __pos = 0) const [noexcept]
```

Find position of a different character.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |



**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find\_first\_not\_of() [2/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_first\\_not\\_of\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_first\\_not\\_of\(\)](#).

**find\_first\_not\_of() [3/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_traits, _Alloc, _Base >::find_first_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search from.                    |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find\_first\_not\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline]
```

Find position of a character not in C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
    _CharT __c,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

**find\_first\_of()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__str</code> | String containing characters to locate. |
|--------------------|-----------------------------------------|

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__pos</code> | Index of character to search from (default 0). |
|--------------------|------------------------------------------------|

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\\_first\\_of\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\\_first\\_of\(\)](#).

#### **find\_first\_of()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const
```

Find position of a character of C substring.

#### Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | String containing characters to locate.    |
| <code>__pos</code> | Index of character to search from.         |
| <code>__n</code>   | Number of characters from s to search for. |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

#### **find\_first\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline]
```

Find position of a character of C string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_not_of (
    _CharT __c,
    size_type __pos = npos) const [noexcept]
```

Find last position of a different character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [size\(\)](#).

**find\_last\_not\_of()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\\_last\\_not\\_of\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\\_last\\_not\\_of\(\)](#).

**find\_last\_not\_of()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const
```

Find last position of a character not in C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find\_last\_not\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline]
```

Find last position of a character not in C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
    _CharT __c,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

**find\_last\_of()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
    const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\\_last\\_of\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\\_last\\_of\(\)](#).

**find\_last\_of()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::size_type \_\_gnu\_cxx::\_\_versa\_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const
```

Find last position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.  |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find\_last\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**front()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front () const [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

**front()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front () [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

**get\_allocator()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::get_allocator ()
const [inline], [noexcept]
```

Return copy of allocator used to construct this string.

**insert()** [1/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    const_iterator __p,
    _CharT __c) [inline]
```

Insert one character.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [2/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    const_iterator __p,
    _InputIterator __beg,
    _InputIterator __end) [inline]
```

Insert a range of characters.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__p</code>   | Const_iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                             |
| <code>__end</code> | End of range.                                               |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [3/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    const_iterator __p,
    size_type __n,
    _CharT __c) [inline]
```

Insert multiple characters.

**Parameters**

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__p</code> | Const_iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                              |



|                                                                                                                   |                          |
|-------------------------------------------------------------------------------------------------------------------|--------------------------|
| <a href="#"></a>                 | The character to insert. |
| <a href="#"></a> <code>_c</code> |                          |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown. Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::insert\(\)](#).

**insert()** [4/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    const_iterator __p,
    std::initializer_list< _CharT > __l) [inline]
```

Insert an initializer\_list of characters.

**Parameters**

|                                                                                                                      |                                                             |
|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| <a href="#"></a> <code>__p</code> | Const_iterator referencing location in string to insert at. |
| <a href="#"></a> <code>__l</code> | The initializer_list of characters to insert.               |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

**insert()** [5/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos,
    const _CharT * __s) [inline]
```

Insert a C string.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__pos</code> | Iterator referencing location in string to insert at. |
| <code>__s</code>   | The C string to insert.                               |

## Returns

Reference to this string.

## Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [6/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos,
    const _CharT * __s,
    size_type __n) [inline]
```

Insert a C substring.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__pos</code> | Iterator referencing location in string to insert at. |
| <code>__s</code>   | The C string to insert.                               |
| <code>__n</code>   | The number of characters to insert.                   |

## Returns

Reference to this string.

## Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [7/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos,
    size_type __n,
    _CharT __c) [inline]
```

Insert multiple characters.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [8/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos1,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Insert value of a string.

**Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>__pos1</code> | Iterator referencing location in string to insert at. |
| <code>__str</code>  | The string to insert.                                 |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [9/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos1,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos2,
    size_type __n) [inline]
```

Insert a substring.

**Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>__pos1</code> | Iterator referencing location in string to insert at. |
| <code>__str</code>  | The string to insert.                                 |
| <code>__pos2</code> | Start of characters in str to insert.                 |
| <code>__n</code>    | Number of characters to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                                               |
|--------------------------------|-------------------------------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                               |
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**length()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length () const [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

**max\_size()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size () const [inline],
[noexcept]
```

Returns the `size()` of the largest possible string.

Referenced by [std::getline\(\)](#).

**operator+=()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (  
    _CharT __c)    [inline]
```

Append a character.

## Parameters

|                 |                          |
|-----------------|--------------------------|
| <code>_↵</code> | The character to append. |
| <code>_C</code> |                          |

## Returns

Reference to this string.

**operator+=()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+= (
    const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Append a string to this string.

## Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

## Returns

Reference to this string.

**operator+=()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+= (
    const _CharT * __s) [inline]
```

Append a C string.

## Parameters

|                 |                         |
|-----------------|-------------------------|
| <code>_↵</code> | The C string to append. |
| <code>_S</code> |                         |

## Returns

Reference to this string.

**operator+=()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+= (
    std::initializer_list<_CharT> __l) [inline]
```

Append an `initializer_list` of characters.

**Parameters**

|   |                                                    |
|---|----------------------------------------------------|
| ↩ | The initializer_list of characters to be appended. |
| ↩ |                                                    |
| ↩ |                                                    |
| ↩ |                                                    |
| / |                                                    |

**Returns**

Reference to this string.

**operator=()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
    __versa_string< _CharT, _Traits, _Alloc, _Base > && __str) [inline], [noexcept]
```

String move assignment operator.

**Parameters**

|       |                |
|-------|----------------|
| __str | Source string. |
|-------|----------------|

The contents of \_\_str are moved into this string (without copying). \_\_str is a valid, but unspecified string.

**operator=()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
    _CharT __c) [inline]
```

Set value to string of length 1.

**Parameters**

|    |                   |
|----|-------------------|
| ↩  | Source character. |
| _c |                   |

Assigning to a character makes this string length 1 and (\*this)[0] == \_\_c.

**operator=()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Assign the value of str to this string.

**Parameters**

|       |                |
|-------|----------------|
| __str | Source string. |
|-------|----------------|

**operator=()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (
    const _CharT * __s) [inline]
```

Copy contents of `__s` into this string.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__s</code> | Source null-terminated string. |
|------------------|--------------------------------|

**operator=()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (
    std::initializer_list<_CharT > __l) [inline]
```

Set value to string constructed from initializer list.

**Parameters**

|                                    |  |
|------------------------------------|--|
| <code>std::initializer_list</code> |  |
|------------------------------------|--|

**operator[]()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[] (
    size_type __pos) const [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Referenced by `__gnu_cxx::__versa_string<char >::back()`, `__gnu_cxx::__versa_string<char >::back()`, `__gnu_cxx::__versa_string<char >::front()` and `__gnu_cxx::__versa_string<char >::front()`.



**operator[]()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
    size_type __pos) [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

**pop\_back()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::pop_back () [inline]
```

Remove the last character.

The string must be non-empty.

**push\_back()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::push_back (
    _CharT __c) [inline]
```

Append a single character.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | Character to append. |
|------------------|----------------------|

Referenced by [\\_\\_gnu\\_cxx::operator+\(\)](#), [\\_\\_gnu\\_cxx::operator+\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::operator+=\(\(\)\)](#).

**rbegin()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::rbegin ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**rbegin()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rbegin () [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

#### **rend()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rend () const
[inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

#### **rend()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rend () [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

#### **replace()** [1/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace (
    const_iterator __i1,
    const_iterator __i2,
    _InputIterator __k1,
    _InputIterator __k2) [inline]
```

Replace range of characters with range.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [2/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Replace range of characters with string.

**Parameters**

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [3/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    const _CharT * __s) [inline]
```

Replace range of characters with C string.

**Parameters**

|                         |                                                 |
|-------------------------|-------------------------------------------------|
| <code>↔<br/>__i1</code> | Iterator referencing start of range to replace. |
| <code>↔<br/>__i2</code> | Iterator referencing end of range to replace.   |
| <code>↔<br/>__s</code>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [4/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    const _CharT * __s,
    size_type __n)    [inline]
```

Replace range of characters with C substring.

**Parameters**

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.     |
| <code>__i2</code> | Iterator referencing end of range to replace.       |
| <code>__s</code>  | C string value to insert.                           |
| <code>__n</code>  | Number of characters from <code>s</code> to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, the first `n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [5/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    size_type __n,
    _CharT __c)    [inline]
```

Replace range of characters with multiple characters.

**Parameters**

|                                                                                                                        |                                                 |
|------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <a href="#"></a><br><code>__i1</code> | Iterator referencing start of range to replace. |
| <a href="#"></a><br><code>__i2</code> | Iterator referencing end of range to replace.   |
| <a href="#"></a><br><code>__n</code>  | Number of characters to insert.                 |
| <a href="#"></a><br><code>__c</code>  | Character to insert.                            |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace() [6/11]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    std::initializer_list< _CharT > __l) [inline]
```

Replace range of characters with `initializer_list`.

**Parameters**

|                                                                                                                          |                                                            |
|--------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <a href="#"></a><br><code>__i1</code> | Iterator referencing start of range to replace.            |
| <a href="#"></a><br><code>__i2</code> | Iterator referencing end of range to replace.              |
| <a href="#"></a><br><code>__l</code>  | The <code>initializer_list</code> of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [7/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace (
    size_type __pos,
    size_type __n,
    const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]
```

Replace characters with value from another string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by `__gnu_cxx::__versa_string<char>::append()`, `__gnu_cxx::__versa_string<char>::assign()`, `__gnu_cxx::__versa_string<char>::insert()`, `__gnu_cxx::__versa_string<char>::insert()`, `__gnu_cxx::__versa_string<char>::insert()`, `__gnu_cxx::__versa_string<char>::insert()`, `__gnu_cxx::__versa_string<char>::insert()`, `__gnu_cxx::__versa_string<char>::replace()`, `__gnu_cxx::__versa_string<char>::replace()`, `__gnu_cxx::__versa_string<char>::replace()`, `__gnu_cxx::__versa_string<char>::replace()`, `__gnu_cxx::__versa_string<char>::replace()`, `__gnu_cxx::__versa_string<char>::replace()`, and `__gnu_cxx::__versa_string<char>::replace()`.

**replace()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace (
    size_type __pos,
    size_type __n1,
    const _CharT * __s) [inline]
```

Replace characters with value of a C string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>std::out_of_range</i> | If <code>__pos &gt; size()</code> .             |
| <i>std::length_error</i> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2) [inline]
```

Replace characters with value of a C substring.

**Parameters**

|                    |                                                    |
|--------------------|----------------------------------------------------|
| <code>__pos</code> | Index of first character to replace.               |
| <code>__n1</code>  | Number of characters to be replaced.               |
| <code>__s</code>   | C string to insert.                                |
| <code>__n2</code>  | Number of characters from <code>__s</code> to use. |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>std::out_of_range</i> | If <code>__pos1 &gt; size()</code> .            |
| <i>std::length_error</i> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    size_type __pos,
    size_type __n1,
    size_type __n2,
    _CharT __c) [inline]
```

Replace characters with multiple characters.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> .             |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace (
    size_type __pos1,
    size_type __n1,
    const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str,
    size_type __pos2,
    size_type __n2) [inline]
```

Replace characters with value from another string.

## Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos1</code> | Index of first character to replace.    |
| <code>__n1</code>   | Number of characters to be replaced.    |
| <code>__str</code>  | String to insert.                       |
| <code>__pos2</code> | Index of first character of str to use. |
| <code>__n2</code>   | Number of characters from str to use.   |

## Returns

Reference to this string.

## Exceptions

|                                |                                                                             |
|--------------------------------|-----------------------------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                             |

Removes the characters in the range `[pos1, pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.



**reserve()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve (
    size_type __res_arg = 0) [inline]
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

**Exceptions**

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Referenced by [\\_\\_gnu\\_cxx::operator+\(\)](#), [\\_\\_gnu\\_cxx::operator+\(\)](#), [\\_\\_gnu\\_cxx::operator+\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::shrink](#)

**resize() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (
    size_type __n) [inline]
```

Resizes the string to the specified number of characters.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
|------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

**resize() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (
    size_type __n,
    _CharT __c)
```

Resizes the string to the specified number of characters.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
|------------------|-------------------------------------------------|

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | Character to fill any new elements. |
|------------------|-------------------------------------|

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

References [append\(\)](#), and [size\(\)](#).

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::resize\(\)](#).

#### **rfind()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string<_CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string<_CharT, ↵
_Traits, _Alloc, _Base >::rfind (
    _CharT __c,
    size_type __pos = npos) const [noexcept]
```

Find last position of a character.

##### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

##### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

#### **rfind()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind (
    const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a string.

##### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

##### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\\_last\\_of\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::rfind\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::rfind\(\)](#).

**rfind()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::rfind (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const
```

Find last position of a C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string to locate.                        |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [std::min\(\)](#), [npos](#), and [size\(\)](#).

**rfind()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (
    const _CharT * __s,
    size_type __pos = npos) const [inline]
```

Find last position of a C string.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**shrink\_to\_fit()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit () [inline],
[noexcept]
```

A non-binding request to reduce capacity() to size().

**size()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size () const [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Referenced by `__gnu_cxx::__versa_string<char>::append()`, `__gnu_cxx::__versa_string<char>::append()`, `__gnu_cxx::__versa_string<char>::assign()`, `__gnu_cxx::__versa_string<char>::assign()`, `__gnu_cxx::__versa_string<char>::assign()`, `__gnu_cxx::__versa_string<char>::at()`, `__gnu_cxx::__versa_string<char>::at()`, `__gnu_cxx::__versa_string<char>::back()`, `__gnu_cxx::__versa_string<char>::back()`, `__gnu_cxx::__versa_string<char>::cend()`, `compare()`, `compare()`, `__gnu_cxx::__versa_string<char>::compare()`, `__gnu_cxx::__versa_string<char>::empty()`, `__gnu_cxx::__versa_string<char>::end()`, `__gnu_cxx::__versa_string<char>::end()`, `find()`, `find()`, `__gnu_cxx::__versa_string<char>::find_first_not_of()`, `find_first_not_of()`, `__gnu_cxx::__versa_string<char>::find_first_not_of()`, `find_first_of()`, `__gnu_cxx::__versa_string<char>::find_last_not_of()`, `find_last_not_of()`, `find_last_of()`, `__gnu_cxx::__versa_string<char>::find_last_of()`, `__gnu_cxx::__versa_string<char>::operator+()`, `__gnu_cxx::__versa_string<char>::operator+()`, `__gnu_cxx::__versa_string<char>::operator+()`, `__gnu_cxx::__versa_string<char>::operator+()`, `__gnu_cxx::__versa_string<char>::operator+()`, `__gnu_cxx::__versa_string<char>::operator==()`, `__gnu_cxx::__versa_string<char>::operator==()`, `__gnu_cxx::__versa_string<char>::operator[]()`, `__gnu_cxx::__versa_string<char>::pop_back()`, `__gnu_cxx::__versa_string<char>::push_back()`, `__gnu_cxx::__versa_string<char>::replace()`, `resize()`, `rfind()`, `rfind()`, and `__gnu_cxx::__versa_string<char>::shrink_to_fit()`.

**substr()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::substr (
    size_type __pos = 0,
    size_type __n = npos) const [inline]
```

Get a substring.

**Parameters**

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

**Returns**

The new string.

**Exceptions**

|                                |                                   |
|--------------------------------|-----------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> . |
|--------------------------------|-----------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

**swap()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap (
    __versa_string< _CharT, _Traits, _Alloc, _Base > & __s) [inline], [noexcept]
```

Swap contents with another string.

## Parameters

|                    |                      |
|--------------------|----------------------|
| <a href="#">_↔</a> | String to swap with. |
| <a href="#">_s</a> |                      |

Exchanges the contents of this string with that of [\\_\\_s](#) in constant time.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::assign\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::operator=\(\)](#), and [\\_\\_gnu\\_cxx::swap\(\)](#).

#### 5.58.4 Member Data Documentation

##### npos

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
const size_type \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >::npos [static]
```

Value returned by various member functions when they fail.

Referenced by [find\(\)](#), [find\(\)](#), [find\\_first\\_not\\_of\(\)](#), [find\\_first\\_not\\_of\(\)](#), [find\\_first\\_of\(\)](#), [find\\_last\\_not\\_of\(\)](#), [find\\_last\\_of\(\)](#), [rfind\(\)](#), and [rfind\(\)](#).

The documentation for this class was generated from the following files:

- [vstring.h](#)
- [vstring.tcc](#)

#### 5.59 \_\_gnu\_debug::\_After\_nth\_from< \_Iterator > Class Template Reference

```
#include <safe_sequence.h>
```

##### Public Member Functions

- [\\_After\\_nth\\_from](#) (const difference\_type &\_\_n, const \_Iterator &[\\_\\_base](#))
- bool [operator\(\)](#) (const \_Iterator &\_\_x) const

##### 5.59.1 Detailed Description

```
template<typename _Iterator>
```

```
class \_\_gnu\_debug::\_After\_nth\_from< \_Iterator >
```

A function object that returns true when the given random access iterator is at least n steps away from the given iterator.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

#### 5.60 std::\_Base\_bitset< \_Nw > Struct Template Reference

```
#include <bitset>
```

##### Public Types

- typedef unsigned long [\\_WordT](#)

##### Public Member Functions

- constexpr [\\_Base\\_bitset](#) (unsigned long long \_\_val) noexcept
- template<size\_t \_Nb>  
constexpr bool [\\_M\\_are\\_all](#) () const noexcept

- constexpr void `_M_do_and` (const `_Base_bitset< _Nw > &__x`) noexcept
- constexpr size\_t `_M_do_count` () const noexcept
- constexpr size\_t `_M_do_find_first` (size\_t) const noexcept
- constexpr size\_t `_M_do_find_next` (size\_t, size\_t) const noexcept
- constexpr void `_M_do_flip` () noexcept
- constexpr void `_M_do_left_shift` (size\_t \_\_shift) noexcept
- constexpr void `_M_do_or` (const `_Base_bitset< _Nw > &__x`) noexcept
- constexpr void `_M_do_reset` () noexcept
- constexpr void `_M_do_right_shift` (size\_t \_\_shift) noexcept
- constexpr void `_M_do_set` () noexcept
- constexpr unsigned long long `_M_do_to_ullong` () const
- constexpr unsigned long `_M_do_to_ulong` () const
- constexpr void `_M_do_xor` (const `_Base_bitset< _Nw > &__x`) noexcept
- constexpr const \_WordT \* `_M_getdata` () const noexcept
- constexpr \_WordT `_M_getword` (size\_t \_\_pos) const noexcept
- constexpr \_WordT & `_M_getword` (size\_t \_\_pos) noexcept
- constexpr \_WordT `_M_hiword` () const noexcept
- constexpr \_WordT & `_M_hiword` () noexcept
- constexpr bool `_M_is_any` () const noexcept
- constexpr bool `_M_is_equal` (const `_Base_bitset< _Nw > &__x`) const noexcept

### Static Public Member Functions

- static constexpr \_WordT `_S_maskbit` (size\_t \_\_pos) noexcept
- static constexpr size\_t `_S_whichbit` (size\_t \_\_pos) noexcept
- static constexpr size\_t `_S_whichbyte` (size\_t \_\_pos) noexcept
- static constexpr size\_t `_S_whichword` (size\_t \_\_pos) noexcept

### Public Attributes

- \_WordT `_M_w` [`_Nw`]

#### 5.60.1 Detailed Description

`template<size_t _Nw>`  
`struct std::_Base_bitset< _Nw >`

Base class, general case. It is a class invariant that `_Nw` will be nonnegative.  
 See documentation for `bitset`.

#### 5.60.2 Member Data Documentation

##### `_M_w`

`template<size_t _Nw>`  
`_WordT std::_Base_bitset< _Nw >::_M_w [_Nw]`

0 is the least significant word.

The documentation for this struct was generated from the following file:

- `bitset`

## 5.61 `std::_Base_bitset< 0 >` Struct Reference

```
#include <bitset>
```

## Public Types

- typedef unsigned long **\_WordT**
- typedef unsigned long **\_WordT**

## Public Member Functions

- constexpr **\_Base\_bitset** (unsigned long long \_\_val) noexcept
- constexpr **\_Base\_bitset** (unsigned long long) noexcept
- constexpr bool **\_M\_are\_all** () const noexcept
- template<size\_t \_Nb>  
constexpr bool **\_M\_are\_all** () const noexcept
- constexpr void **\_M\_do\_and** (const **\_Base\_bitset**< 0 > &) noexcept
- constexpr void **\_M\_do\_and** (const **\_Base\_bitset**< \_Nw > &\_\_x) noexcept
- constexpr size\_t **\_M\_do\_count** () const noexcept
- constexpr size\_t **\_M\_do\_count** () const noexcept
- constexpr size\_t **\_M\_do\_find\_first** (size\_t) const noexcept
- constexpr size\_t **\_M\_do\_find\_first** (size\_t) const noexcept
- constexpr size\_t **\_M\_do\_find\_next** (size\_t, size\_t) const noexcept
- constexpr size\_t **\_M\_do\_find\_next** (size\_t, size\_t) const noexcept
- constexpr void **\_M\_do\_flip** () noexcept
- constexpr void **\_M\_do\_flip** () noexcept
- constexpr void **\_M\_do\_left\_shift** (size\_t \_\_shift) noexcept
- constexpr void **\_M\_do\_left\_shift** (size\_t) noexcept
- constexpr void **\_M\_do\_or** (const **\_Base\_bitset**< 0 > &) noexcept
- constexpr void **\_M\_do\_or** (const **\_Base\_bitset**< \_Nw > &\_\_x) noexcept
- constexpr void **\_M\_do\_reset** () noexcept
- constexpr void **\_M\_do\_reset** () noexcept
- constexpr void **\_M\_do\_right\_shift** (size\_t \_\_shift) noexcept
- constexpr void **\_M\_do\_right\_shift** (size\_t) noexcept
- constexpr void **\_M\_do\_set** () noexcept
- constexpr void **\_M\_do\_set** () noexcept
- constexpr unsigned long long **\_M\_do\_to\_ullong** () const
- constexpr unsigned long long **\_M\_do\_to\_ullong** () const noexcept
- constexpr unsigned long **\_M\_do\_to\_ulong** () const
- constexpr unsigned long **\_M\_do\_to\_ulong** () const noexcept
- constexpr void **\_M\_do\_xor** (const **\_Base\_bitset**< 0 > &) noexcept
- constexpr void **\_M\_do\_xor** (const **\_Base\_bitset**< \_Nw > &\_\_x) noexcept
- constexpr const \_WordT \* **\_M\_getdata** () const noexcept
- constexpr \_WordT **\_M\_getword** (size\_t \_\_pos) const noexcept
- constexpr \_WordT & **\_M\_getword** (size\_t \_\_pos) noexcept
- constexpr \_WordT **\_M\_getword** (size\_t) const noexcept
- \_WordT & **\_M\_getword** (size\_t) noexcept
- constexpr \_WordT **\_M\_hiword** () const noexcept
- constexpr \_WordT **\_M\_hiword** () const noexcept
- constexpr \_WordT & **\_M\_hiword** () noexcept
- constexpr bool **\_M\_is\_any** () const noexcept
- constexpr bool **\_M\_is\_any** () const noexcept
- constexpr bool **\_M\_is\_equal** (const **\_Base\_bitset**< 0 > &) const noexcept
- constexpr bool **\_M\_is\_equal** (const **\_Base\_bitset**< \_Nw > &\_\_x) const noexcept

### Static Public Member Functions

- static constexpr `_WordT _S_maskbit (size_t __pos)` noexcept
- static constexpr `_WordT _S_maskbit (size_t __pos)` noexcept
- static constexpr `size_t _S_whichbit (size_t __pos)` noexcept
- static constexpr `size_t _S_whichbit (size_t __pos)` noexcept
- static constexpr `size_t _S_whichbyte (size_t __pos)` noexcept
- static constexpr `size_t _S_whichbyte (size_t __pos)` noexcept
- static constexpr `size_t _S_whichword (size_t __pos)` noexcept
- static constexpr `size_t _S_whichword (size_t __pos)` noexcept

### Public Attributes

- `_WordT _M_w [_Nw]`

#### 5.61.1 Detailed Description

Base class, specialization for no storage (zero-length bitset).  
See documentation for `bitset`.

#### 5.61.2 Member Data Documentation

##### `_M_w`

`_WordT std::_Base_bitset< _Nw >::_M_w[_Nw]`

0 is the least significant word.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.62 `std::_Base_bitset< 1 >` Struct Reference

```
#include <bitset>
```

### Public Types

- typedef unsigned long `_WordT`
- typedef unsigned long `_WordT`

### Public Member Functions

- constexpr `_Base_bitset (unsigned long long __val)` noexcept
- constexpr `_Base_bitset (unsigned long long __val)` noexcept
- constexpr `bool _M_are_all ()` const noexcept
- template<size\_t \_Nb>  
constexpr `bool _M_are_all ()` const noexcept
- constexpr void `_M_do_and (const _Base_bitset< 1 > &__x)` noexcept
- constexpr void `_M_do_and (const _Base_bitset< _Nw > &__x)` noexcept
- constexpr `size_t _M_do_count ()` const noexcept
- constexpr `size_t _M_do_count ()` const noexcept
- constexpr `size_t _M_do_find_first (size_t __not_found)` const noexcept
- constexpr `size_t _M_do_find_first (size_t)` const noexcept
- constexpr `size_t _M_do_find_next (size_t __prev, size_t __not_found)` const noexcept
- constexpr `size_t _M_do_find_next (size_t, size_t)` const noexcept
- constexpr void `_M_do_flip ()` noexcept



- constexpr void **\_M\_do\_flip** () noexcept
- constexpr void **\_M\_do\_left\_shift** (size\_t \_\_shift) noexcept
- constexpr void **\_M\_do\_left\_shift** (size\_t \_\_shift) noexcept
- constexpr void **\_M\_do\_or** (const [\\_Base\\_bitset](#)< 1 > &\_\_x) noexcept
- constexpr void **\_M\_do\_or** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) noexcept
- constexpr void **\_M\_do\_reset** () noexcept
- constexpr void **\_M\_do\_reset** () noexcept
- constexpr void **\_M\_do\_right\_shift** (size\_t \_\_shift) noexcept
- constexpr void **\_M\_do\_right\_shift** (size\_t \_\_shift) noexcept
- constexpr void **\_M\_do\_set** () noexcept
- constexpr void **\_M\_do\_set** () noexcept
- constexpr unsigned long long **\_M\_do\_to\_ullong** () const
- constexpr unsigned long long **\_M\_do\_to\_ullong** () const noexcept
- constexpr unsigned long **\_M\_do\_to\_ulong** () const
- constexpr unsigned long **\_M\_do\_to\_ulong** () const noexcept
- constexpr void **\_M\_do\_xor** (const [\\_Base\\_bitset](#)< 1 > &\_\_x) noexcept
- constexpr void **\_M\_do\_xor** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) noexcept
- constexpr const \_WordT \* **\_M\_getdata** () const noexcept
- constexpr const \_WordT \* **\_M\_getdata** () const noexcept
- constexpr \_WordT **\_M\_getword** (size\_t \_\_pos) const noexcept
- constexpr \_WordT & **\_M\_getword** (size\_t \_\_pos) noexcept
- constexpr \_WordT **\_M\_getword** (size\_t) const noexcept
- constexpr \_WordT & **\_M\_getword** (size\_t) noexcept
- constexpr \_WordT **\_M\_hiword** () const noexcept
- constexpr \_WordT **\_M\_hiword** () const noexcept
- constexpr \_WordT & **\_M\_hiword** () noexcept
- constexpr \_WordT & **\_M\_hiword** () noexcept
- constexpr bool **\_M\_is\_any** () const noexcept
- constexpr bool **\_M\_is\_any** () const noexcept
- constexpr bool **\_M\_is\_equal** (const [\\_Base\\_bitset](#)< 1 > &\_\_x) const noexcept
- constexpr bool **\_M\_is\_equal** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) const noexcept

### Static Public Member Functions

- static constexpr \_WordT **\_S\_maskbit** (size\_t \_\_pos) noexcept
- static constexpr \_WordT **\_S\_maskbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbyte** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbyte** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichword** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichword** (size\_t \_\_pos) noexcept

### Public Attributes

- \_WordT **\_M\_w** [\_Nw]
- \_WordT **\_M\_w**

### 5.62.1 Detailed Description

Base class, specialization for a single word.  
See documentation for bitset.

### 5.62.2 Member Data Documentation

#### `_M_w`

`_WordT std::_Base_bitset<_Nw>::_M_w[_Nw]`

0 is the least significant word.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.63 `__gnu_debug::_BeforeBeginHelper<_Sequence>` Struct Template Reference

```
#include <safe_iterator.h>
```

### Static Public Member Functions

- `template<typename _Iterator, typename _Category>`  
`static bool _S_Is (const \_Safe\_iterator<_Iterator, _Sequence, _Category> &)`
- `template<typename _Iterator, typename _Category>`  
`static bool _S_Is_Beginnest (const \_Safe\_iterator<_Iterator, _Sequence, _Category> &__it)`

#### 5.63.1 Detailed Description

```
template<typename _Sequence>
struct __gnu_debug::_BeforeBeginHelper<_Sequence>
```

Helper struct to deal with sequence offering a `before_begin` iterator.

The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

## 5.64 `std::_Bind<_Signature>` Class Template Reference

### 5.64.1 Detailed Description

```
template<typename _Signature>
class std::_Bind<_Signature>
```

Type of the function object returned from `bind()`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.65 `std::_Bind_result<_Result, _Signature>` Class Template Reference

### 5.65.1 Detailed Description

```
template<typename _Result, typename _Signature>
class std::_Bind_result<_Result, _Signature>
```

Type of the function object returned from `bind<R>()`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.66 `__gnu_cxx::__detail::_Bitmap_counter<_Tp>` Class Template Reference

```
#include <bitmap_allocator.h>
```

## Public Member Functions

- [\\_Bitmap\\_counter](#) ([\\_BPVector](#) &Rvbp, long \_\_index=-1)
- pointer [\\_M\\_base](#) () const throw ()
- bool [\\_M\\_finished](#) () const throw ()
- std::size\_t \* [\\_M\\_get](#) () const throw ()
- [\\_Index\\_type](#) [\\_M\\_offset](#) () const throw ()
- void [\\_M\\_reset](#) (long \_\_index=-1) throw ()
- void [\\_M\\_set\\_internal\\_bitmap](#) (std::size\_t \* \_\_new\_internal\_marker) throw ()
- [\\_Index\\_type](#) [\\_M\\_where](#) () const throw ()
- [\\_Bitmap\\_counter](#) & [operator++](#) () throw ()

### 5.66.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::_Bitmap_counter< _Tp >
```

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.67 std::\_\_detail::\_BracketMatcher< \_TraitsT, \_\_icase, \_\_collate > Struct Template Reference

```
#include <regex_compiler.h>
```

## Public Types

- typedef [\\_TraitsT::char\\_class\\_type](#) [\\_CharClassT](#)
- typedef [\\_TransT:: \\_CharT](#) [\\_CharT](#)
- typedef [\\_TraitsT::string\\_type](#) [\\_StringT](#)
- typedef [\\_TransT:: \\_StrTransT](#) [\\_StrTransT](#)
- typedef [\\_RegexTranslator< \\_TraitsT, \\_\\_icase, \\_\\_collate >](#) [\\_TransT](#)

## Public Member Functions

- [\\_BracketMatcher](#) (bool \_\_is\_non\_matching, const [\\_TraitsT](#) &\_\_traits)
- void [\\_M\\_add\\_char](#) ([\\_CharT](#) \_\_c)
- void [\\_M\\_add\\_character\\_class](#) (const [\\_StringT](#) &\_\_s, bool \_\_neg)
- [\\_StringT](#) [\\_M\\_add\\_collate\\_element](#) (const [\\_StringT](#) &\_\_s)
- void [\\_M\\_add\\_equivalence\\_class](#) (const [\\_StringT](#) &\_\_s)
- void [\\_M\\_make\\_range](#) ([\\_CharT](#) \_\_l, [\\_CharT](#) \_\_r)
- void [\\_M\\_ready](#) ()
- bool [operator\(\)](#) ([\\_CharT](#) \_\_ch) const

### 5.67.1 Detailed Description

```
template<typename _TraitsT, bool __icase, bool __collate>
struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >
```

Matches a character range (bracket expression)

The documentation for this struct was generated from the following files:

- [regex\\_compiler.h](#)
- [regex\\_compiler.tcc](#)

## 5.68 `__gnu_cxx::_Caster<_ToType>` Struct Template Reference

```
#include <cast.h>
```

### Public Types

- `typedef _ToType::element_type * type`

#### 5.68.1 Detailed Description

```
template<typename _ToType>
struct __gnu_cxx::_Caster<_ToType>
```

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

The documentation for this struct was generated from the following file:

- [cast.h](#)

## 5.69 `__gnu_cxx::_Char_types<_CharT>` Struct Template Reference

```
#include <char_traits.h>
```

### Public Types

- `typedef unsigned long int_type`
- `typedef std::streamoff off_type`
- `typedef std::streampos pos_type`
- `typedef std::mbstate_t state_type`

#### 5.69.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::_Char_types<_CharT>
```

Mapping from character type to associated types.

#### Note

This is an implementation class for the generic version of `char_traits`. It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, streamoff, streampos, and mbstate\_t. Users who need a different set of types, but who don't need to change the definitions of any function defined in `char_traits`, can specialize `__gnu_cxx::_Char_types` while leaving `__gnu_cxx::char_traits` alone.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.70 `__gnu_pbds::detail::pat_trie_base::_CIter<Node, Leaf, Head, Inode, Is_Forward_Iterator>` Class Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



## Public Types

- typedef allocator\_type **\_Alloc**
- typedef Node::allocator\_type **allocator\_type**
- typedef type\_traits::const\_pointer **const\_pointer**
- typedef type\_traits::const\_reference **const\_reference**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef [rebind\\_traits](#)< \_Alloc, Head >::pointer **head\_pointer**
- typedef Inode::iterator **inode\_iterator**
- typedef [rebind\\_traits](#)< \_Alloc, Inode >::pointer **inode\_pointer**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits](#)< \_Alloc, Leaf >::const\_pointer **leaf\_const\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Leaf >::pointer **leaf\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Node >::pointer **node\_pointer**
- typedef type\_traits::pointer **pointer**
- typedef type\_traits::reference **reference**
- typedef Node::type\_traits **type\_traits**
- typedef type\_traits::value\_type **value\_type**

## Public Member Functions

- **\_CIter** (const [\\_CIter](#)< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- **\_CIter** (node\_pointer p\_nd=0)
- bool **operator!=** (const [\\_CIter](#) &other) const
- bool **operator!=** (const [\\_CIter](#)< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const
- const\_reference **operator\*** () const

- `_Clter` & `operator++` ()
- `_Clter` `operator++` (int)
- `_Clter` & `operator--` ()
- `_Clter` `operator--` (int)
- `const_pointer` `operator->` () const
- `_Clter` & `operator=` (const `_Clter` &other)
- `_Clter` & `operator=` (const `_Clter`< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- `bool` `operator==` (const `_Clter` &other) const
- `bool` `operator==` (const `_Clter`< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const

### Public Attributes

- `node_pointer` `m_p_nd`

### Protected Member Functions

- `void` `dec` (false\_type)
- `void` `dec` (true\_type)
- `void` `inc` (false\_type)
- `void` `inc` (true\_type)

### Static Protected Member Functions

- `static` `node_pointer` `get_larger_sibling` (`node_pointer` p\_nd)
- `static` `node_pointer` `get_smaller_sibling` (`node_pointer` p\_nd)
- `static` `leaf_pointer` `leftmost_descendant` (`node_pointer` p\_nd)
- `static` `leaf_pointer` `rightmost_descendant` (`node_pointer` p\_nd)

#### 5.70.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Const iterator.

The documentation for this class was generated from the following file:

- `pat_trie_base.hpp`

## 5.71 `std::__detail::__Compiler<_TraitsT>` Class Template Reference

```
#include <regex_compiler.h>
```

### Public Types

- `typedef` `_TraitsT::char_type` `_CharT`
- `typedef` `regex_constants::syntax_option_type` `_FlagT`
- `typedef` `_NFA<_TraitsT>` `_RegexT`

### Public Member Functions

- `_Compiler` (const `_CharT` \*\_\_b, const `_CharT` \*\_\_e, const `typename` `_TraitsT::locale_type` &\_\_traits, `_FlagT` \_\_flags)
- `shared_ptr`< const `_RegexT` > `_M_get_nfa` () noexcept

### 5.71.1 Detailed Description

```
template<typename _TraitsT>
class std::__detail::_Compiler< _TraitsT >
```

Builds an NFA from an input iterator range.

The `_TraitsT` type should fulfill requirements [28.3].

The documentation for this class was generated from the following files:

- [regex\\_compiler.h](#)
- [regex\\_compiler.tcc](#)

## 5.72 `std::__parallel::_CRandNumber< _MustBeInt >` Struct Template Reference

```
#include <algo.h>
```

### Public Member Functions

- `int operator() (int __limit)`

### 5.72.1 Detailed Description

```
template<typename _MustBeInt = int>
struct std::__parallel::_CRandNumber< _MustBeInt >
```

Functor wrapper for `std::rand()`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

## 5.73 `std::_Deque_base< _Tp, _Alloc >` Class Template Reference

```
#include <stl_deque.h>
```

Inheritance diagram for `std::_Deque_base< _Tp, _Alloc >`:



### Protected Types

- `enum { _S_initial_map_size }`
- `typedef __gnu_cxx::__alloc_traits< _Tp_alloc_type > _Alloc_traits`
- `typedef __gnu_cxx::__alloc_traits< _Map_alloc_type > _Map_alloc_traits`

- typedef \_Alloc\_traits::template rebind<\_Ptr>::other **\_Map\_alloc\_type**
- typedef iterator::\_Map\_pointer **\_Map\_pointer**
- typedef \_Alloc\_traits::pointer **\_Ptr**
- typedef \_Alloc\_traits::const\_pointer **\_Ptr\_const**
- typedef \_\_gnu\_cxx::\_\_alloc\_traits<\_Alloc>::template rebind<\_Tp>::other **\_Tp\_alloc\_type**
- typedef \_Alloc **allocator\_type**
- typedef [\\_Deque\\_iterator](#)<\_Tp, const \_Tp &, \_Ptr\_const> **const\_iterator**
- typedef [\\_Deque\\_iterator](#)<\_Tp, \_Tp &, \_Ptr> **iterator**

### Protected Member Functions

- [\\_Deque\\_base](#) ([\\_Deque\\_base](#) &&\_\_x)
- [\\_Deque\\_base](#) ([\\_Deque\\_base](#) &&\_\_x, const allocator\_type &\_\_a)
- [\\_Deque\\_base](#) ([\\_Deque\\_base](#) &&\_\_x, const allocator\_type &\_\_a, size\_t \_\_n)
- [\\_Deque\\_base](#) (const allocator\_type &\_\_a)
- [\\_Deque\\_base](#) (const allocator\_type &\_\_a, size\_t \_\_num\_elements)
- [\\_Deque\\_base](#) (size\_t \_\_num\_elements)
- \_Map\_pointer **\_M\_allocate\_map** (size\_t \_\_n)
- \_Ptr **\_M\_allocate\_node** ()
- void **\_M\_create\_nodes** (\_Map\_pointer \_\_nstart, \_Map\_pointer \_\_nfinish)
- void **\_M\_deallocate\_map** (\_Map\_pointer \_\_p, size\_t \_\_n) noexcept
- void **\_M\_deallocate\_node** (\_Ptr \_\_p) noexcept
- void **\_M\_destroy\_nodes** (\_Map\_pointer \_\_nstart, \_Map\_pointer \_\_nfinish) noexcept
- \_Map\_alloc\_type **\_M\_get\_map\_allocator** () const noexcept
- const \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- void [\\_M\\_initialize\\_map](#) (size\_t)
- allocator\_type **get\_allocator** () const noexcept

### Protected Attributes

- [\\_Deque\\_impl](#) **\_M\_impl**

#### 5.73.1 Detailed Description

template<typename \_Tp, typename \_Alloc>

class std::Deque\_base<\_Tp, \_Alloc>

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier. Nothing in this class ever constructs or destroys an actual Tp element. (Deque handles that itself.) Only/All memory management is performed here.

#### 5.73.2 Member Function Documentation

##### [\\_M\\_initialize\\_map\(\)](#)

```
template<typename _Tp, typename _Alloc>
void std::Deque_base<_Tp, _Alloc>::_M_initialize_map (
    size_t __num_elements) [protected]
```

Layout storage.

##### Parameters

|                             |                                                        |
|-----------------------------|--------------------------------------------------------|
| <code>__num_elements</code> | The count of T's for which to allocate space at first. |
|-----------------------------|--------------------------------------------------------|



## Returns

Nothing.

The initial underlying memory layout is a bit complicated...

References [std::max\(\)](#).

The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

## 5.74 std::Deque\_iterator<\_Tp, \_Ref, \_Ptr > Struct Template Reference

```
#include <stl_deque.h>
```

### Public Types

- typedef [\\_\\_ptr\\_rebind](#)< \_Ptr, \_Tp > **\_Elt\_pointer**
- typedef [\\_\\_ptr\\_rebind](#)< \_Ptr, \_Elt\_pointer > **\_Map\_pointer**
- typedef [\\_Deque\\_iterator](#) **\_Self**
- typedef [\\_\\_iter](#)< const \_Tp > **const\_iterator**
- typedef ptrdiff\_t **difference\_type**
- typedef [\\_\\_iter](#)< \_Tp > **iterator**
- typedef [std::random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Ptr **pointer**
- typedef \_Ref **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- [\\_Deque\\_iterator](#) (\_Elt\_pointer \_\_x, \_Map\_pointer \_\_y) noexcept
- [\\_Deque\\_iterator](#) (const [\\_Deque\\_iterator](#) &\_\_x) noexcept
- template<typename \_Iter, typename = \_Require<is\_same<\_Self, const\_iterator>, is\_same<\_Iter, iterator>>>  
  [\\_Deque\\_iterator](#) (const \_Iter &\_\_x) noexcept
- [iterator\\_M\\_const\\_cast](#) () const noexcept
- void [\\_M\\_set\\_node](#) (\_Map\_pointer \_\_new\_node) noexcept
- reference **operator\*** () const noexcept
- [\\_Self](#) & **operator++** () noexcept
- [\\_Self](#) **operator++** (int) noexcept
- [\\_Self](#) & **operator+=** (difference\_type \_\_n) noexcept
- [\\_Self](#) & **operator--** () noexcept
- [\\_Self](#) **operator--** (int) noexcept
- [\\_Self](#) & **operator-=** (difference\_type \_\_n) noexcept
- pointer **operator->** () const noexcept
- [\\_Deque\\_iterator](#) & **operator=** (const [\\_Deque\\_iterator](#) &)=default
- reference **operator[]** (difference\_type \_\_n) const noexcept

### Static Public Member Functions

- static size\_t **\_S\_buffer\_size** () noexcept

## Public Attributes

- `_Elt_pointer _M_cur`
- `_Elt_pointer _M_first`
- `_Elt_pointer _M_last`
- `_Map_pointer _M_node`

## Friends

- `_Self operator+ (const _Self &__x, difference_type __n) noexcept`
- `_Self operator+ (difference_type __n, const _Self &__x) noexcept`
- `template<typename _RefR, typename _PtrR>  
difference_type operator- (const _Self &__x, const _Deque_iterator<_Tp, _RefR, _PtrR> &__y) noexcept`
- `difference_type operator- (const _Self &__x, const _Self &__y) noexcept`
- `_Self operator- (const _Self &__x, difference_type __n) noexcept`
- `strong_ordering operator<=> (const _Self &__x, const _Self &__y) noexcept`
- `template<typename _RefR, typename _PtrR>  
bool operator== (const _Self &__x, const _Deque_iterator<_Tp, _RefR, _PtrR> &__y) noexcept`
- `bool operator== (const _Self &__x, const _Self &__y) noexcept`

### 5.74.1 Detailed Description

`template<typename _Tp, typename _Ref, typename _Ptr>  
struct std::_Deque_iterator<_Tp, _Ref, _Ptr>`

A deque::iterator.

Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

### 5.74.2 Member Function Documentation

#### `_M_set_node()`

```
template<typename _Tp, typename _Ref, typename _Ptr>
void std::_Deque_iterator<_Tp, _Ref, _Ptr>::_M_set_node (
    _Map_pointer __new_node) [inline], [noexcept]
```

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

The documentation for this struct was generated from the following files:

- `stl_algobase.h`
- `stl_deque.h`

## 5.75 `__gnu_parallel::__DRandomShufflingGlobalData<_RAIter>` Struct Template Reference

```
#include <random_shuffle.h>
```

## Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::iterator_traits<_RAIter> _TraitsType`
- `typedef _TraitsType::value_type _ValueType`

## Public Member Functions

- [\\_DRandomShufflingGlobalData](#) ([\\_RAIter](#) & [\\_\\_source](#))

## Public Attributes

- [\\_ThreadIndex](#) \* [\\_M\\_bin\\_proc](#)
- [\\_DifferenceType](#) \*\* [\\_M\\_dist](#)
- [int](#) [\\_M\\_num\\_bins](#)
- [int](#) [\\_M\\_num\\_bits](#)
- [\\_RAIter](#) & [\\_M\\_source](#)
- [\\_DifferenceType](#) \* [\\_M\\_starts](#)
- [\\_ValueType](#) \*\* [\\_M\\_temporaries](#)

### 5.75.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >
```

Data known to every thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).

### 5.75.2 Constructor & Destructor Documentation

#### [\\_DRandomShufflingGlobalData\(\)](#)

```
template<typename _RAIter>
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_DRandomShufflingGlobalData (
    _RAIter & __source) [inline]
```

Constructor.

References [\\_M\\_source](#).

### 5.75.3 Member Data Documentation

#### [\\_M\\_bin\\_proc](#)

```
template<typename _RAIter>
_ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_bin_proc
```

Number of the thread that will further process the corresponding bin.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

#### [\\_M\\_dist](#)

```
template<typename _RAIter>
_DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist
```

Two-dimensional array to hold the thread-bin distribution.

Dimensions ([\\_M\\_num\\_threads](#) + 1) \_\_x ([\\_M\\_num\\_bins](#) + 1).

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

#### [\\_M\\_num\\_bins](#)

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins
```

Number of bins to distribute to.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

### `_M_num_bits`

```
template<typename _RAIter>
```

```
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits
```

Number of bits needed to address the bins.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

### `_M_source`

```
template<typename _RAIter>
```

```
_RAIter& __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_source
```

Begin iterator of the `_source`.

Referenced by [\\_DRandomShufflingGlobalData\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

### `_M_starts`

```
template<typename _RAIter>
```

```
_DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts
```

Start indexes of the threads' `_chunks`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

### `_M_temporaries`

```
template<typename _RAIter>
```

```
_ValueType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries
```

Temporary arrays for each thread.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## 5.76 `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference

```
#include <random_shuffle.h>
```

### Public Attributes

- [\\_BinIndex \\_\\_bins\\_end](#)
- [\\_BinIndex \\_M\\_bins\\_begin](#)
- [int \\_M\\_num\\_threads](#)
- [\\_DRandomShufflingGlobalData<\\_RAIter> \\* \\_M\\_sd](#)
- [uint32\\_t \\_M\\_seed](#)

#### 5.76.1 Detailed Description

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >
```

Local data for a thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).

#### 5.76.2 Member Data Documentation

##### `__bins_end`

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
_BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end
```

End index for bins taken care of by this thread.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#).

### **`_M_bins_begin`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
\_BinIndex \_\_gnu\_parallel::\_\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_bins\_begin
```

Begin index for bins taken care of by this thread.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

### **`_M_num_threads`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
int \_\_gnu\_parallel::\_\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_num\_threads
```

Number of threads participating in total.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

### **`_M_sd`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
\_DRandomShufflingGlobalData<\_RAIter>\* \_\_gnu\_parallel::\_\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_sd
```

Pointer to global data.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

### **`_M_seed`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
uint32\_t \_\_gnu\_parallel::\_\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_seed
```

Random `_M_seed` for this thread.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#).

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## **5.77 `__gnu_parallel::__DummyReduct` Struct Reference**

```
#include <for_each_selectors.h>
```

### **Public Member Functions**

- `bool operator() (bool, bool) const`

#### **5.77.1 Detailed Description**

Reduction function doing nothing.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## **5.78 `__gnu_debug::__Equal_to< _Type >` Class Template Reference**

```
#include <safe_sequence.h>
```

**Public Member Functions**

- `_Equal_to` (const `_Type` &\_\_v)
- bool `operator()` (const `_Type` &\_\_x) const

**5.78.1 Detailed Description**

```
template<typename _Type>
class __gnu_debug::_Equal_to<_Type >
```

A simple function object that returns true if the passed-in value is equal to the stored value.  
The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

**5.79 `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>` Class Template Reference**

```
#include <base.h>
```

Inheritance diagram for `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>`:

**Public Types**

- typedef `_T1` [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

**Public Member Functions**

- `_EqualFromLess` (`_Compare` &\_\_comp)
- bool `operator()` (const `_T1` &\_\_a, const `_T2` &\_\_b)

**5.79.1 Detailed Description**

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_EqualFromLess<_T1, _T2, _Compare >
```

Constructs predicate for equality from strict weak ordering predicate.

## 5.79.2 Member Typedef Documentation

### first\_argument\_type

```
typedef _T1 std::binary\_function< _T1, _T2, bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

### result\_type

```
typedef bool std::binary\_function< _T1, _T2, bool >::result_type [inherited]
```

`result_type` is the return type

### second\_argument\_type

```
typedef _T2 std::binary\_function< _T1, _T2, bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

The documentation for this class was generated from the following file:

- [base.h](#)

## 5.80 \_\_gnu\_parallel::\_EqualTo< \_T1, \_T2 > Struct Template Reference

```
#include <base.h>
```

Inheritance diagram for `__gnu_parallel::_EqualTo< _T1, _T2 >`:



### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### Public Member Functions

- `bool` **operator()** (const `_T1` &\_\_t1, const `_T2` &\_\_t2) const

### 5.80.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::__EqualTo< _T1, _T2 >
```

Similar to std::equal\_to, but allows two different types.

### 5.80.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
typedef bool std::binary_function< _T1, _T2, bool >::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [base.h](#)

## 5.81 std::\_\_detail::\_\_Executor< \_Bilter, \_Alloc, \_TraitsT, \_\_dfs\_mode > Class Template Reference

```
#include <regex_executor.h>
```

### Public Types

- typedef [iterator\\_traits](#)< \_Bilter >::value\_type **\_CharT**
- typedef \_TraitsT::char\_class\_type **\_ClassT**
- typedef [regex\\_constants::match\\_flag\\_type](#) **\_FlagT**
- typedef \_NFA< \_TraitsT > **\_NFAT**
- typedef [basic\\_regex](#)< \_CharT, \_TraitsT > **\_RegexT**
- typedef ::vector< [sub\\_match](#)< \_Bilter >, \_Alloc > **\_ResultsVec**

### Public Member Functions

- **\_Executor** (\_Bilter \_\_begin, \_Bilter \_\_end, [\\_ResultsVec](#) &\_\_results, const [\\_RegexT](#) &\_\_re, [\\_FlagT](#) \_\_flags)
- bool **\_M\_match** ()
- bool **\_M\_search** ()
- bool **\_M\_search\_from\_first** ()

### Public Attributes

- \_Bilter **\_M\_begin**
- [\\_ResultsVec](#) **\_M\_cur\_results**
- \_Bilter **\_M\_current**
- const \_Bilter **\_M\_end**
- [\\_FlagT](#) **\_M\_flags**
- bool **\_M\_has\_sol**



- `const _NFAT & _M_nfa`
- `const _RegexT & _M_re`
- `::vector< pair< _Bilter, int > > _M_rep_count`
- `_ResultsVec & _M_results`
- `_State_info< __search_mode, _ResultsVec > _M_states`

### 5.81.1 Detailed Description

```
template<typename _Bilter, typename _Alloc, typename _TraitsT, bool __dfs_mode>
class std::__detail::_Executor< _Bilter, _Alloc, _TraitsT, __dfs_mode >
```

Takes a regex and an input string and does the matching.

The `_Executor` class has two modes: DFS mode and BFS mode, controlled by the template parameter `__dfs_mode`.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex\\_executor.h](#)
- [regex\\_executor.tcc](#)

## 5.82 \_\_gnu\_cxx::\_ExtPtr\_allocator< \_Tp > Class Template Reference

```
#include <extptr_allocator.h>
```

### Public Types

- `typedef _Pointer_adapter< _Relative_pointer_impl< const _Tp > > const_pointer`
- `typedef const _Tp & const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Pointer_adapter< _Relative_pointer_impl< _Tp > > pointer`
- `typedef _Tp & reference`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

### Public Member Functions

- `_ExtPtr_allocator (const _ExtPtr_allocator &__rarg) noexcept`
- `template<typename _Up>`  
`_ExtPtr_allocator (const _ExtPtr_allocator< _Up > &__rarg) noexcept`
- `const std::allocator< _Tp > & _M_getUnderlyingImp () const`
- `const_pointer address (const_reference __x) const noexcept`
- `pointer address (reference __x) const noexcept`
- `pointer allocate (size_type __n, const void *__p=0)`
- `template<typename _Up, typename... _Args>`  
`void construct (_Up *__p, _Args &&... __args)`
- `template<typename... _Args>`  
`void construct (pointer __p, _Args &&... __args)`
- `void deallocate (pointer __p, size_type __n)`
- `template<typename _Up>`  
`void destroy (_Up *__p)`
- `void destroy (pointer __p)`
- `size_type max_size () const noexcept`
- `bool operator== (const _ExtPtr_allocator &__rarg) const`
- `template<typename _Up>`  
`bool operator== (const _ExtPtr_allocator< _Up > &__rarg) const`

**Friends**

- `template<typename _Up>`  
`void swap (\_ExtPtr\_allocator<_Up> &, \_ExtPtr\_allocator<_Up> &)`

**5.82.1 Detailed Description**

```
template<typename _Tp>
class __gnu_cxx::_ExtPtr_allocator<_Tp>
```

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See `ext/pointer.h`) Memory allocation in this example is still performed using `std::allocator`.

The documentation for this class was generated from the following file:

- [extptr\\_allocator.h](#)

**5.83 `__gnu_cxx::__detail::_Ffit_finder<_Tp>` Class Template Reference**

```
#include <bitmap_allocator.h>
```

**Public Types**

- typedef [\\_Block\\_pair](#) `argument_type`
- typedef bool `result_type`

**Public Member Functions**

- `std::size_t * \_M\_get () const throw ()`
- `_Counter_type \_M\_offset () const throw ()`
- `bool operator() (\_Block\_pair __bp) throw ()`

**5.83.1 Detailed Description**

```
template<typename _Tp>
class __gnu_cxx::__detail::_Ffit_finder<_Tp>
```

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

**5.84 `std::_Function_base` Class Reference**

```
#include <std_function.h>
```

Inheritance diagram for `std::_Function_base`:



### Public Types

- `using _Manager_type`

### Public Member Functions

- `bool _M_empty () const`

### Public Attributes

- `_Any_data _M_functor`
- `_Manager_type _M_manager`

### Static Public Attributes

- `static const size_t _M_max_align`
- `static const size_t _M_max_size`

#### 5.84.1 Detailed Description

Base class of all polymorphic function object wrappers.

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

## 5.85 `std::_Fwd_list_base<_Tp, _Alloc >` Struct Template Reference

```
#include <forward_list.h>
```

Inheritance diagram for std::\_Fwd\_list\_base< \_Tp, \_Alloc >:



### Public Types

- using **const\_iterator**
- using **iterator**
- using **pointer**

### Public Member Functions

- `_Fwd_list_base ( _Fwd_list_base && )=default`
- `_Fwd_list_base ( _Fwd_list_base && __lst, _Node_alloc_type && __a )`
- `_Fwd_list_base ( _Fwd_list_base && __lst, _Node_alloc_type && __a, std::true_type )`
- `_Fwd_list_base ( _Node_alloc_type && __a )`
- `const _Node_alloc_type & _M_get_Node_allocator () const noexcept`
- `_Node_alloc_type & _M_get_Node_allocator () noexcept`
- `template<typename... _Args>`  
`auto _M_insert_after (const_iterator __pos, _Args &&... __args) -> _Base_ptr`

### Protected Types

- using **\_Base\_ptr**
- using **\_Node**
- using **\_Node\_alloc\_traits**
- using **\_Node\_alloc\_type**
- using **\_Node\_ptr**
- using **\_Node\_traits**

### Protected Member Functions

- `template<typename... _Args>`  
`_Node_ptr _M_create_node ( _Args &&... __args )`
- `void _M_destroy_node ( _Node_ptr __p )`
- `_Base_ptr _M_erase_after ( _Base_ptr __pos )`
- `_Base_ptr _M_erase_after ( _Base_ptr __pos, _Base_ptr __last )`
- `_Node * _M_get_node ()`

- `template<typename... _Args>  
_Base_ptr M_insert_after (const_iterator __pos, _Args &&... __args)`
- `void M_put_node (_Node_ptr __p)`

### Protected Attributes

- `_Fwd_list_impl M_impl`

### 5.85.1 Detailed Description

```
template<typename _Tp, typename _Alloc>  
struct std::_Fwd_list_base<_Tp, _Alloc >
```

Base class for `forward_list`.

The documentation for this struct was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 5.86 `std::_Fwd_list_const_iterator<_Tp >` Struct Template Reference

```
#include <forward_list.h>
```

### Public Types

- `typedef const \_Fwd\_list\_node<_Tp > _Node`
- `typedef \_Fwd\_list\_const\_iterator<_Tp > _Self`
- `typedef ptrdiff_t difference_type`
- `typedef \_Fwd\_list\_iterator<_Tp > iterator`
- `typedef std::forward\_iterator\_tag iterator_category`
- `typedef const _Tp * pointer`
- `typedef const _Tp & reference`
- `typedef _Tp value_type`

### Public Member Functions

- `_Fwd_list_const_iterator (const \_Fwd\_list\_node\_base *__n) noexcept`
- `_Fwd_list_const_iterator (const iterator &__iter) noexcept`
- `reference operator* () const noexcept`
- `_Self & operator++ () noexcept`
- `_Self operator++ (int) noexcept`
- `pointer operator-> () const noexcept`

### Friends

- `template<typename, typename>  
struct _Fwd_list_base`
- `template<typename, typename>  
class forward_list`
- `bool operator== (const _Self &__x, const _Self &__y) noexcept`

### 5.86.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_const_iterator< _Tp >
```

A forward\_list::const\_iterator.

All the functions are op overloads.

### 5.86.2 Friends And Related Symbol Documentation

#### operator==

```
template<typename _Tp>
bool operator== (
    const _Self & __x,
    const _Self & __y) [friend]
```

Forward list const\_iterator equality comparison.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.87 std::\_Fwd\_list\_iterator< \_Tp > Struct Template Reference

```
#include <forward_list.h>
```

### Public Types

- typedef [\\_Fwd\\_list\\_node](#)< \_Tp > **\_Node**
- typedef [\\_Fwd\\_list\\_iterator](#)< \_Tp > **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

### Public Member Functions

- **\_Fwd\_list\_iterator** ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_n) noexcept
- reference **operator\*** () const noexcept
- [\\_Self](#) & **operator++** () noexcept
- [\\_Self](#) **operator++** (int) noexcept
- pointer **operator->** () const noexcept

### Friends

- template<typename, typename>  
struct **\_Fwd\_list\_base**
- struct **\_Fwd\_list\_const\_iterator**< \_Tp >
- template<typename, typename>  
class **forward\_list**
- bool **operator==** (const [\\_Self](#) & \_\_x, const [\\_Self](#) & \_\_y) noexcept

### 5.87.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_iterator< _Tp >
```

A forward\_list::iterator.

All the functions are op overloads.

### 5.87.2 Friends And Related Symbol Documentation

#### operator==

```
template<typename _Tp>
bool operator== (
    const _Self & __x,
    const _Self & __y) [friend]
```

Forward list iterator equality comparison.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.88 std::\_Fwd\_list\_node< \_Tp > Struct Template Reference

```
#include <forward_list.h>
```

Inheritance diagram for std::\_Fwd\_list\_node< \_Tp >:



#### Public Types

- using [\\_Base\\_ptr](#)
- using [\\_Node\\_ptr](#)

#### Public Member Functions

- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_base\\_ptr](#) ()
- const [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_base\\_ptr](#) () const
- [\\_Node\\_ptr](#) [\\_M\\_node\\_ptr](#) ()
- void [\\_M\\_reverse\\_after](#) () noexcept
- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_transfer\\_after](#) ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_begin, [\\_Fwd\\_list\\_node\\_base](#) \* \_\_end) noexcept

- `const _Tp * _M_valptr ()` `const noexcept`
- `_Tp * _M_valptr ()` `noexcept`

#### Public Attributes

- `_Fwd_list_node_base * _M_next`
- `__gnu_cxx::__aligned_buffer< _Tp > _M_storage`

#### 5.88.1 Detailed Description

`template<typename _Tp>`  
`struct std::_Fwd_list_node< _Tp >`

A helper node class for `forward_list`. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.89 std::\_Fwd\_list\_node\_base Struct Reference

`#include <forward_list.h>`

Inheritance diagram for `std::_Fwd_list_node_base`:



#### Public Types

- `using _Base_ptr`

#### Public Member Functions

- `_Fwd_list_node_base (_Fwd_list_node_base &&__x)` `noexcept`
- `_Fwd_list_node_base (const _Fwd_list_node_base &)=delete`
- `_Fwd_list_node_base * _M_base_ptr ()`
- `const _Fwd_list_node_base * _M_base_ptr ()` `const`
- `void _M_reverse_after ()` `noexcept`
- `_Fwd_list_node_base * _M_transfer_after (_Fwd_list_node_base * __begin, _Fwd_list_node_base * __end)` `noexcept`
- `_Fwd_list_node_base & operator= (_Fwd_list_node_base &&__x)` `noexcept`
- `_Fwd_list_node_base & operator= (const _Fwd_list_node_base &)=delete`



## Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_next](#)

### 5.89.1 Detailed Description

A helper basic node class for `forward_list`.

```
This is just a linked list with nothing inside it.
There are purely list shuffling utility methods here.
```

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.90 \_\_gnu\_parallel::\_GuardedIterator<\_RAIter, \_Compare > Class Template Reference

```
#include <multiway_merge.h>
```

### Public Member Functions

- [\\_GuardedIterator](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [\\_Compare](#) & \_\_comp)
- [operator \\_RAIter](#) () const
- [std::iterator\\_traits](#)< [\\_RAIter](#) >::value\_type & [operator\\*](#) () const
- [\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > & [operator++](#) ()

### Friends

- bool [operator<](#) (const [\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > & \_\_bi1, const [\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > & \_\_bi2)
- bool [operator<=](#) (const [\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > & \_\_bi1, const [\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > & \_\_bi2)

### 5.90.1 Detailed Description

```
template<typename _RAIter, typename _Compare>
class __gnu_parallel::_GuardedIterator<_RAIter, _Compare >
```

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

The implicit supremum comes with a performance cost.

Deriving from `_RAIter` is not possible since `_RAIter` need not be a class.

### 5.90.2 Constructor & Destructor Documentation

#### `_GuardedIterator()`

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator<_RAIter, _Compare >::_GuardedIterator (
    _RAIter __begin,
    _RAIter __end,
    _Compare & __comp) [inline]
```

Constructor. Sets iterator to beginning of sequence.

#### Parameters

|                      |                             |
|----------------------|-----------------------------|
| <code>__begin</code> | Begin iterator of sequence. |
| <code>__end</code>   | End iterator of sequence.   |

|                     |                                                                  |
|---------------------|------------------------------------------------------------------|
| <code>__comp</code> | Comparator provided for associated overloaded compare operators. |
|---------------------|------------------------------------------------------------------|

Referenced by [operator<](#), and [operator<=](#).

### 5.90.3 Member Function Documentation

#### **operator \_RAIter()**

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator _RAIter () const [inline]
```

Convert to wrapped iterator.

##### Returns

Wrapped iterator.

#### **operator\*()**

```
template<typename _RAIter, typename _Compare>
std::iterator_traits< _RAIter >::value_type & __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator* () const [inline]
```

Dereference operator.

##### Returns

Referenced element.

#### **operator++()**

```
template<typename _RAIter, typename _Compare>
__GuardedIterator< _RAIter, _Compare > & __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator++ () [inline]
```

Pre-increment operator.

##### Returns

This.

### 5.90.4 Friends And Related Symbol Documentation

#### **operator<**

```
template<typename _RAIter, typename _Compare>
bool operator< (
    const __GuardedIterator< _RAIter, _Compare > & __bi1,
    const __GuardedIterator< _RAIter, _Compare > & __bi2) [friend]
```

Compare two elements referenced by guarded iterators.

##### Parameters

|                    |                  |
|--------------------|------------------|
| <code>__bi1</code> | First iterator.  |
| <code>__bi2</code> | Second iterator. |

##### Returns

true if less.

References [\\_GuardedIterator\(\)](#).

**operator<=**

```
template<typename _RAIter, typename _Compare>
bool operator<= (
    const _GuardedIterator< _RAIter, _Compare > & __bi1,
    const _GuardedIterator< _RAIter, _Compare > & __bi2) [friend]
```

Compare two elements referenced by guarded iterators.

**Parameters**

|                    |                  |
|--------------------|------------------|
| <code>__bi1</code> | First iterator.  |
| <code>__bi2</code> | Second iterator. |

**Returns**

True if less equal.

References [\\_GuardedIterator\(\)](#).

The documentation for this class was generated from the following file:

- [multiway\\_merge.h](#)

## 5.91 `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata >` Struct Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata >`:

**Public Types**

- typedef `_ATraits::const_iterator` **a\_const\_iterator**
- typedef `detail::rebind_traits<_Alloc, _ATraits >::const_pointer` **a\_const\_pointer**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**

- typedef [\\_Node\\_base](#)<\_ATraits, Metadata > **base\_type**
- typedef base\_type::node\_pointer **node\_pointer**
- typedef base\_type::type\_traits **type\_traits**

### Public Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_min**
- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

#### 5.91.1 Detailed Description

template<typename \_ATraits, typename Metadata>  
struct \_\_gnu\_pbds::detail::pat\_trie\_base::\_Head<\_ATraits, Metadata >

Head node for PATRICIA tree.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.92 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode<\_ATraits, Metadata > Struct Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode<\_ATraits, Metadata >:



### Classes

- struct [const\\_iterator](#)
- struct [iterator](#)

## Public Types

- enum { **arr\_size** }
- typedef [detail::rebind\\_traits](#)< \_Alloc, node\_pointer > **\_\_rebind\_np**
- typedef base\_type::allocator\_type **\_Alloc**
- typedef base\_type::access\_traits **access\_traits**
- typedef \_Alloc **allocator\_type**
- typedef [\\_Node\\_base](#)< \_ATraits, Metadata > **base\_type**
- typedef \_\_rebind\_np::pointer **node\_pointer\_pointer**
- typedef \_\_rebind\_np::reference **node\_pointer\_reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef base\_type::type\_traits **type\_traits**
- typedef type\_traits::value\_type **value\_type**

## Public Member Functions

- **\_Inode** (size\_type, const a\_const\_iterator)
- node\_pointer **add\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- [iterator](#) **begin** ()
- [const\\_iterator](#) **begin** () const
- [iterator](#) **end** ()
- [const\\_iterator](#) **end** () const
- [iterator](#) **get\_child\_it** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_const\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer) const
- size\_type **get\_e\_ind** () const
- node\_const\_pointer **get\_join\_child** (node\_const\_pointer, a\_const\_pointer) const
- node\_pointer **get\_join\_child** (node\_pointer, a\_const\_pointer)
- node\_pointer **get\_lower\_bound\_child\_node** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer)
- leaf\_pointer **leftmost\_descendant** ()
- leaf\_const\_pointer **leftmost\_descendant** () const
- a\_const\_iterator **pref\_b\_it** () const
- a\_const\_iterator **pref\_e\_it** () const
- void **remove\_child** ([iterator](#))
- void **remove\_child** (node\_pointer)
- void **replace\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- leaf\_pointer **rightmost\_descendant** ()
- leaf\_const\_pointer **rightmost\_descendant** () const
- bool **should\_be\_mine** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer) const
- void **update\_prefixes** (a\_const\_pointer)

## Public Attributes

- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

### 5.92.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >
```

Internal node type, PATRICIA tree.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.93 \_\_gnu\_cxx::\_Invalid\_type Struct Reference

```
#include <pointer.h>
```

### 5.93.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of `void*`, `const void*`, and so on.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 5.94 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Iter< Node, Leaf, Head, Inode, Is\_Forward\_Iterator > Class Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



### Public Types

- `typedef allocator_type _Alloc`
- `typedef base_type::allocator_type allocator_type`
- `typedef \_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > base_type`
- `typedef type_traits::const_pointer const_pointer`
- `typedef type_traits::const_reference const_reference`
- `typedef allocator_type::difference_type difference_type`
- `typedef base_type::head_pointer head_pointer`
- `typedef Inode::iterator inode_iterator`
- `typedef base_type::inode_pointer inode_pointer`
- `typedef std::bidirectional\_iterator\_tag iterator_category`
- `typedef base_type::leaf_const_pointer leaf_const_pointer`
- `typedef base_type::leaf_pointer leaf_pointer`
- `typedef base_type::node_pointer node_pointer`
- `typedef type_traits::pointer pointer`
- `typedef type_traits::reference reference`
- `typedef base_type::type_traits type_traits`
- `typedef type_traits::value_type value_type`

## Public Member Functions

- `_Iter` (const `_Iter`< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- `_Iter` (node\_pointer p\_nd=0)
- bool `operator!=` (const `_CIter` &other) const
- bool `operator!=` (const `_CIter`< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const
- reference `operator*` () const
- `_Iter` & `operator++` ()
- `_Iter` `operator++` (int)
- `_Iter` & `operator--` ()
- `_Iter` `operator--` (int)
- pointer `operator->` () const
- `_Iter` & `operator=` (const `_Iter` &other)
- `_Iter` & `operator=` (const `_Iter`< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- bool `operator==` (const `_CIter` &other) const
- bool `operator==` (const `_CIter`< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const

## Public Attributes

- node\_pointer `m_p_nd`

## Protected Member Functions

- void `dec` (false\_type)
- void `dec` (true\_type)
- void `inc` (false\_type)
- void `inc` (true\_type)

## Static Protected Member Functions

- static node\_pointer `get_larger_sibling` (node\_pointer p\_nd)
- static node\_pointer `get_smaller_sibling` (node\_pointer p\_nd)
- static leaf\_pointer `leftmost_descendant` (node\_pointer p\_nd)
- static leaf\_pointer `rightmost_descendant` (node\_pointer p\_nd)

### 5.94.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Iterator.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.95 std::\_\_fwlist::\_Iterator< \_Const, \_Ptr > Class Template Reference

```
#include <forward_list.h>
```

## Public Types

- using `difference_type`
- using `iterator_category`
- using `pointer`
- using `reference`
- using `value_type`

## Public Member Functions

- `constexpr _Iterator` (`_Base_ptr __x`) `noexcept`
- `_Iterator` (`const _Iterator &`)=default
- `constexpr _Iterator` (`const _Iterator< false, _Ptr > &__i`)
- `constexpr reference operator*` (`()`) `const noexcept`
- `constexpr _Iterator & operator++` (`()`) `noexcept`
- `constexpr _Iterator operator++` (`int`) `noexcept`
- `constexpr pointer operator->` (`()`) `const noexcept`
- `_Iterator & operator=` (`const _Iterator &`)=default

## Friends

- `template<typename _Tp, typename _Allocator>`  
`struct ::Fwd_list_base`
- `template<typename _Tp, typename _Allocator>`  
`class ::forward_list`
- `constexpr bool operator==` (`const _Iterator &__x, const _Iterator &__y`) `noexcept`

### 5.95.1 Detailed Description

`template<bool _Const, typename _Ptr>`  
`class std::__fwlist::_Iterator<_Const, _Ptr>`

A `forward_list` iterator when the allocator uses fancy pointers.

### 5.95.2 Friends And Related Symbol Documentation

#### `operator==`

```
template<bool _Const, typename _Ptr>
bool operator== (
    const _Iterator< _Const, _Ptr > & __x,
    const _Iterator< _Const, _Ptr > & __y) [friend]
```

Forward list iterator equality comparison.

The documentation for this class was generated from the following file:

- [forward\\_list.h](#)

## 5.96 `__gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>` Class Template Reference

```
#include <iterator.h>
```



Inheritance diagram for `__gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory >`:



### Public Types

- typedef `std::iterator_traits<_Iterator1 >_TraitsType`
- typedef `_TraitsType::difference_type difference_type`
- typedef `_Iterator1 first_type`
- typedef `_Iterator1 first_type`
- typedef `_Iterator1 first_type`
- typedef `_IteratorCategory iterator_category`
- typedef `_IteratorPair * pointer`
- typedef `_IteratorPair & reference`
- typedef `_Iterator2 second_type`
- typedef `_Iterator2 second_type`
- typedef `_Iterator2 second_type`
- typedef void `value_type`

### Public Member Functions

- `_IteratorPair` (const `_Iterator1` &\_\_first, const `_Iterator2` &\_\_second)
- `operator _Iterator2` () const
- `_IteratorPair operator+` (difference\_type \_\_delta) const
- `_IteratorPair & operator++` ()
- const `_IteratorPair operator++` (int)
- difference\_type `operator-` (const `_IteratorPair` &\_\_other) const
- `_IteratorPair & operator--` ()
- const `_IteratorPair operator--` (int)
- `_IteratorPair & operator=` (const `_IteratorPair` &\_\_other)
- constexpr void `swap` (pair &\_\_p) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_Iterator1 >, \_\_is\_nothrow\_swappable<\_Iterator2 >>::value)
- constexpr void `swap` (pair &\_\_p) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_Iterator1 >, \_\_is\_nothrow\_swappable<\_Iterator2 >>::value)
- constexpr void `swap` (pair &\_\_p) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_Iterator1 >, \_\_is\_nothrow\_swappable<\_Iterator2 >>::value)
- constexpr void `swap` (pair &\_\_p) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_Iterator1 >, \_\_is\_nothrow\_swappable<\_Iterator2 >>::value)

## Public Attributes

- `_Iterator1` [first](#)
- `_Iterator1` [first](#)
- `_Iterator1` [first](#)
- `_Iterator1` [first](#)
- `_Iterator2` [second](#)
- `_Iterator2` [second](#)
- `_Iterator2` [second](#)
- `_Iterator2` [second](#)

## Related Symbols

(Note that these are not member symbols.)

- `constexpr enable_if<__and<__is_swappable<_Iterator1>, __is_swappable<_Iterator2>>::value>::type swap(pair<_Iterator1, _Iterator2> &__x, pair<_Iterator1, _Iterator2> &__y) noexcept(noexcept(__x.swap(↵__y)))`
- `constexpr enable_if<__and<__is_swappable<_Iterator1>, __is_swappable<_Iterator2>>::value>::type swap(pair<_Iterator1, _Iterator2> &__x, pair<_Iterator1, _Iterator2> &__y) noexcept(noexcept(__x.swap(↵__y)))`
- `constexpr enable_if<__and<__is_swappable<_Iterator1>, __is_swappable<_Iterator2>>::value>::type swap(pair<_Iterator1, _Iterator2> &__x, pair<_Iterator1, _Iterator2> &__y) noexcept(noexcept(__x.swap(↵__y)))`
- `constexpr enable_if<__and<__is_swappable<_Iterator1>, __is_swappable<_Iterator2>>::value>::type swap(pair<_Iterator1, _Iterator2> &__x, pair<_Iterator1, _Iterator2> &__y) noexcept(noexcept(__x.swap(↵__y)))`
- `constexpr bool operator==(const pair<_Iterator1, _Iterator2> &__x, const pair<_U1, _U2> &__y)`
- `constexpr bool operator==(const pair<_Iterator1, _Iterator2> &__x, const pair<_U1, _U2> &__y)`
- `constexpr bool operator==(const pair<_Iterator1, _Iterator2> &__x, const pair<_U1, _U2> &__y)`
- `constexpr bool operator==(const pair<_Iterator1, _Iterator2> &__x, const pair<_U1, _U2> &__y)`
- `constexpr common_comparison_category_t<__detail::__synth3way_t<_Iterator1, _U1>, __detail::__synth3way_t<_Iterator2, _U2>> operator<=>(const pair<_Iterator1, _Iterator2> &__x, const pair<_U1, _U2> &__y)`
- `constexpr common_comparison_category_t<__detail::__synth3way_t<_Iterator1, _U1>, __detail::__synth3way_t<_Iterator2, _U2>> operator<=>(const pair<_Iterator1, _Iterator2> &__x, const pair<_U1, _U2> &__y)`
- `constexpr common_comparison_category_t<__detail::__synth3way_t<_Iterator1, _U1>, __detail::__synth3way_t<_Iterator2, _U2>> operator<=>(const pair<_Iterator1, _Iterator2> &__x, const pair<_U1, _U2> &__y)`
- `constexpr common_comparison_category_t<__detail::__synth3way_t<_Iterator1, _U1>, __detail::__synth3way_t<_Iterator2, _U2>> operator<=>(const pair<_Iterator1, _Iterator2> &__x, const pair<_U1, _U2> &__y)`

### 5.96.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>
class __gnu_parallel::IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

### 5.96.2 Member Typedef Documentation

**first\_type** [1/3]

```
typedef _Iterator1 std::pair<_Iterator1, _Iterator2>::first_type [inherited]
The type of the first member.
```

**first\_type** [2/3]

```
typedef _Iterator1 std::pair< _Iterator1, _Iterator2 >::first_type [inherited]
```

The type of the first member.

**first\_type** [3/3]

```
typedef _Iterator1 std::pair< _Iterator1, _Iterator2 >::first_type [inherited]
```

The type of the first member.

**second\_type** [1/3]

```
typedef _Iterator2 std::pair< _Iterator1, _Iterator2 >::second_type [inherited]
```

The type of the second member.

**second\_type** [2/3]

```
typedef _Iterator2 std::pair< _Iterator1, _Iterator2 >::second_type [inherited]
```

The type of the second member.

**second\_type** [3/3]

```
typedef _Iterator2 std::pair< _Iterator1, _Iterator2 >::second_type [inherited]
```

The type of the second member.

### 5.96.3 Member Function Documentation

**swap()** [1/4]

```
void std::pair< _Iterator1, _Iterator2 >::swap (  
    pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
```

Swap the first members and then the second members.

**swap()** [2/4]

```
void std::pair< _Iterator1, _Iterator2 >::swap (  
    pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
```

Swap the first members and then the second members.

**swap()** [3/4]

```
void std::pair< _Iterator1, _Iterator2 >::swap (  
    pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
```

Swap the first members and then the second members.

**swap()** [4/4]

```
void std::pair< _Iterator1, _Iterator2 >::swap (  
    pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
```

Swap the first members and then the second members.

### 5.96.4 Friends And Related Symbol Documentation

**operator<=>()** [1/4]

```
common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__synth3way_t< _Iterator2, _U2 > > operator<=> (  
    __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__synth3way_t< _Iterator2, _U2 >)
```

```
const pair<_Iterator1, _Iterator2> & __x,
const pair<_U1, _U2> & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, P is ordered before Q if P.first is less than Q.first, or if P.first and Q.first are equivalent (neither is less than the other) and P.second is less than Q.second.

#### **operator<=>() [2/4]**

```
common_comparison_category_t<__detail::__synth3way_t<_Iterator1, _U1>, __detail::__synth3way_t<_Iterator2, _U2>> operator<=> (
    const pair<_Iterator1, _Iterator2> & __x,
    const pair<_U1, _U2> & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, P is ordered before Q if P.first is less than Q.first, or if P.first and Q.first are equivalent (neither is less than the other) and P.second is less than Q.second.

#### **operator<=>() [3/4]**

```
common_comparison_category_t<__detail::__synth3way_t<_Iterator1, _U1>, __detail::__synth3way_t<_Iterator2, _U2>> operator<=> (
    const pair<_Iterator1, _Iterator2> & __x,
    const pair<_U1, _U2> & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, P is ordered before Q if P.first is less than Q.first, or if P.first and Q.first are equivalent (neither is less than the other) and P.second is less than Q.second.

#### **operator<=>() [4/4]**

```
common_comparison_category_t<__detail::__synth3way_t<_Iterator1, _U1>, __detail::__synth3way_t<_Iterator2, _U2>> operator<=> (
    const pair<_Iterator1, _Iterator2> & __x,
    const pair<_U1, _U2> & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, P is ordered before Q if P.first is less than Q.first, or if P.first and Q.first are equivalent (neither is less than the other) and P.second is less than Q.second.

#### **operator==() [1/4]**

```
bool operator== (
    const pair<_Iterator1, _Iterator2> & __x,
    const pair<_U1, _U2> & __y) [related]
```

Two pairs are equal iff their members are equal.

#### **operator==() [2/4]**

```
bool operator== (
    const pair<_Iterator1, _Iterator2> & __x,
    const pair<_U1, _U2> & __y) [related]
```

Two pairs are equal iff their members are equal.

#### **operator==() [3/4]**

```
bool operator== (
    const pair<_Iterator1, _Iterator2> & __x,
    const pair<_U1, _U2> & __y) [related]
```

Two pairs are equal iff their members are equal.

#### **operator==()** [4/4]

```
bool operator== (
    const pair< _Iterator1, _Iterator2 > & __x,
    const pair< _U1, _U2 > & __y) [related]
```

Two pairs are equal iff their members are equal.

#### **swap()** [1/4]

```
enable_if< __and< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
    pair< _Iterator1, _Iterator2 > & __x,
    pair< _Iterator1, _Iterator2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

#### Note

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

#### **swap()** [2/4]

```
enable_if< __and< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
    pair< _Iterator1, _Iterator2 > & __x,
    pair< _Iterator1, _Iterator2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

#### Note

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

#### **swap()** [3/4]

```
enable_if< __and< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
    pair< _Iterator1, _Iterator2 > & __x,
    pair< _Iterator1, _Iterator2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

#### Note

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

#### **swap()** [4/4]

```
enable_if< __and< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
    pair< _Iterator1, _Iterator2 > & __x,
    pair< _Iterator1, _Iterator2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

## Note

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

## 5.96.5 Member Data Documentation

**first** [1/4]

`_Iterator1 std::pair<_Iterator1, _Iterator2>::first` [inherited]

The first member.

**first** [2/4]

`_Iterator1 std::pair<_Iterator1, _Iterator2>::first` [inherited]

The first member.

**first** [3/4]

`_Iterator1 std::pair<_Iterator1, _Iterator2>::first` [inherited]

The first member.

**first** [4/4]

`_Iterator1 std::pair<_Iterator1, _Iterator2>::first` [inherited]

The first member.

**second** [1/4]

`_Iterator2 std::pair<_Iterator1, _Iterator2>::second` [inherited]

The second member.

**second** [2/4]

`_Iterator2 std::pair<_Iterator1, _Iterator2>::second` [inherited]

The second member.

**second** [3/4]

`_Iterator2 std::pair<_Iterator1, _Iterator2>::second` [inherited]

The second member.

**second** [4/4]

`_Iterator2 std::pair<_Iterator1, _Iterator2>::second` [inherited]

The second member.

The documentation for this class was generated from the following file:

- [iterator.h](#)

5.97 `__gnu_parallel::IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>` Class Template Reference

```
#include <iterator.h>
```

## Public Types

- typedef [std::iterator\\_traits](#)< [\\_Iterator1](#) >::difference\_type **difference\_type**
- typedef [\\_IteratorCategory](#) **iterator\_category**
- typedef [\\_IteratorTriple](#) \* **pointer**
- typedef [\\_IteratorTriple](#) & **reference**
- typedef void **value\_type**

## Public Member Functions

- [\\_IteratorTriple](#) (const [\\_Iterator1](#) &\_\_first, const [\\_Iterator2](#) &\_\_second, const [\\_Iterator3](#) &\_\_third)
- **operator** [\\_Iterator3](#) () const
- [\\_IteratorTriple](#) **operator**+ (difference\_type \_\_delta) const
- [\\_IteratorTriple](#) & **operator**++ ()
- const [\\_IteratorTriple](#) **operator**++ (int)
- difference\_type **operator**- (const [\\_IteratorTriple](#) &\_\_other) const
- [\\_IteratorTriple](#) & **operator**-- ()
- const [\\_IteratorTriple](#) **operator**-- (int)
- [\\_IteratorTriple](#) & **operator**= (const [\\_IteratorTriple](#) &\_\_other)

## Public Attributes

- [\\_Iterator1](#) **\_M\_first**
- [\\_Iterator2](#) **\_M\_second**
- [\\_Iterator3](#) **\_M\_third**

### 5.97.1 Detailed Description

```
template<typename \_Iterator1, typename \_Iterator2, typename \_Iterator3, typename \_IteratorCategory>
class \_\_gnu\_parallel::\_IteratorTriple< \_Iterator1, \_Iterator2, \_Iterator3, \_IteratorCategory >
```

A triple of iterators. The usual iterator operations are applied to all three child iterators.  
The documentation for this class was generated from the following file:

- [iterator.h](#)

## 5.98 [\\_\\_gnu\\_parallel::\\_Job](#)< [\\_DifferenceTp](#) > Struct Template Reference

```
#include <workstealing.h>
```

## Public Types

- typedef [\\_DifferenceTp](#) **\_DifferenceType**

## Public Attributes

- volatile [\\_DifferenceType](#) [\\_M\\_first](#)
- volatile [\\_DifferenceType](#) [\\_M\\_last](#)
- volatile [\\_DifferenceType](#) [\\_M\\_load](#)

### 5.98.1 Detailed Description

```
template<typename \_DifferenceTp>
struct \_\_gnu\_parallel::\_Job< \_DifferenceTp >
```

One [\\_\\_job](#) for a certain thread.

### 5.98.2 Member Data Documentation

#### `_M_first`

```
template<typename _DifferenceTp>
```

```
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_first
```

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

#### `_M_last`

```
template<typename _DifferenceTp>
```

```
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_last
```

Last element.

Changed by owning thread only.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

#### `_M_load`

```
template<typename _DifferenceTp>
```

```
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_load
```

Number of elements, i.e. `_M_last - _M_first + 1`.

Changed by owning thread only.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

The documentation for this struct was generated from the following file:

- [workstealing.h](#)

## 5.99 `__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata>` Struct Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata>`:





## Public Types

- typedef `_ATraits::const_iterator` **a\_const\_iterator**
- typedef `detail::rebind_traits<_Alloc, _ATraits>::const_pointer` **a\_const\_pointer**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `_Node_base<_ATraits, Metadata>` **base\_type**
- typedef `type_traits::const_reference` **const\_reference**
- typedef `detail::rebind_traits<_Alloc, _Node_base>::pointer` **node\_pointer**
- typedef `type_traits::reference` **reference**
- typedef `base_type::type_traits` **type\_traits**
- typedef `type_traits::value_type` **value\_type**

## Public Member Functions

- `_Leaf` (const\_reference other)
- reference **value** ()
- const\_reference **value** () const

## Public Attributes

- node\_pointer **m\_p\_parent**
- const `node_type` **m\_type**

### 5.99.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata>
```

Leaf node for PATRICIA tree.

The documentation for this struct was generated from the following file:

- `pat_trie_base.hpp`

### 5.100 \_\_gnu\_parallel::\_Less<\_T1, \_T2> Struct Template Reference

```
#include <base.h>
```

Inheritance diagram for `__gnu_parallel::_Less<_T1, _T2>`:



## Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

## Public Member Functions

- `bool operator()` (`const _T1 &__t1, const _T2 &__t2`) `const`
- `bool operator()` (`const _T2 &__t2, const _T1 &__t1`) `const`

### 5.100.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::_Less<_T1, _T2>
```

Similar to `std::less`, but allows two different types.

### 5.100.2 Member Typedef Documentation

#### `first_argument_type`

typedef `_T1` [std::binary\\_function](#)< `_T1`, `_T2`, `bool` >::`first_argument_type` [inherited]  
`first_argument_type` is the type of the first argument

#### `result_type`

typedef `bool` [std::binary\\_function](#)< `_T1`, `_T2`, `bool` >::`result_type` [inherited]  
`result_type` is the return type

#### `second_argument_type`

typedef `_T2` [std::binary\\_function](#)< `_T1`, `_T2`, `bool` >::`second_argument_type` [inherited]  
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [base.h](#)

## 5.101 `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>` Class Template Reference

```
#include <multiseq_selection.h>
```

Inheritance diagram for `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare >`:



### Public Types

- typedef `std::pair<_T1, _T2>` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair<_T1, _T2>` `second_argument_type`

### Public Member Functions

- `_Lexicographic` (`_Compare` & `__comp`)
- `bool operator()` (`const std::pair<_T1, _T2>` & `__p1`, `const std::pair<_T1, _T2>` & `__p2`) `const`

#### 5.101.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_Lexicographic<_T1, _T2, _Compare >
```

Compare \_\_a pair of types lexicographically, ascending.

#### 5.101.2 Member Typedef Documentation

##### `first_argument_type`

```
typedef std::pair<_T1, _T2> std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### `result_type`

```
typedef bool std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool>::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

```
typedef std::pair<_T1, _T2> std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool>::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

## 5.102 `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>` Class Template Reference

```
#include <multiseq_selection.h>
```

Inheritance diagram for `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>`:

**Public Types**

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

**Public Member Functions**

- `_LexicographicReverse` (`_Compare` &`__comp`)
- `bool operator()` (`const std::pair<_T1, _T2> &__p1`, `const std::pair<_T1, _T2> &__p2`) `const`

**5.102.1 Detailed Description**

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>
```

Compare `__a` pair of types lexicographically, descending.

**5.102.2 Member Typedef Documentation****first\_argument\_type**

```
typedef _T1 std::binary_function<_T1, _T2, bool>::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

**result\_type**

```
typedef bool std::binary\_function< _T1, _T2, bool >::result_type [inherited]
```

result\_type is the return type

**second\_argument\_type**

```
typedef _T2 std::binary\_function< _T1, _T2, bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

**5.103 std::\_List\_base< \_Tp, \_Alloc > Class Template Reference**

```
#include <stl_list.h>
```

Inheritance diagram for std::\_List\_base< \_Tp, \_Alloc >:

**Public Types**

- typedef \_Alloc **allocator\_type**

**Public Member Functions**

- **\_List\_base** ([\\_List\\_base](#) &&)=default
- **\_List\_base** ([\\_List\\_base](#) &&\_\_x, [\\_Node\\_alloc\\_type](#) &&\_\_a)
- **\_List\_base** ([\\_Node\\_alloc\\_type](#) &&\_\_a)
- **\_List\_base** ([\\_Node\\_alloc\\_type](#) &&\_\_a, [\\_List\\_base](#) &&\_\_x)
- **\_List\_base** (const [\\_Node\\_alloc\\_type](#) &\_\_a) noexcept
- void **\_M\_clear** () noexcept
- size\_t **\_M\_distance** (const [\\_\\_detail::\\_List\\_node\\_base](#) \* \_\_first, const [\\_\\_detail::\\_List\\_node\\_base](#) \* \_\_last) const
- const [\\_Node\\_alloc\\_type](#) & **\_M\_get\_Node\_allocator** () const noexcept
- [\\_Node\\_alloc\\_type](#) & **\_M\_get\_Node\_allocator** () noexcept
- void **\_M\_init** () noexcept
- void **\_M\_move\_nodes** ([\\_List\\_base](#) &&\_\_x)
- size\_t **\_M\_node\_count** () const

**Static Public Member Functions**

- static `size_t` **S\_distance** (const `__detail::_List_node_base` \* \_\_first, const `__detail::_List_node_base` \* \_\_last)

**Protected Types**

- typedef `__gnu_cxx::__alloc_traits< _Node_alloc_type > _Node_alloc_traits`
- typedef `_Tp_alloc_traits::template rebind< typename _Node_traits::_Node >::other _Node_alloc_type`
- using `_Node_ptr`
- typedef `__list::_Node_traits< _Tp, typename _Tp_alloc_traits::pointer > _Node_traits`
- typedef `__gnu_cxx::__alloc_traits< _Tp_alloc_type > _Tp_alloc_traits`
- typedef `__gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Tp >::other _Tp_alloc_type`

**Protected Member Functions**

- void **\_M\_dec\_size** (size\_t \_\_n)
- void **\_M\_destroy\_node** (\_Node\_ptr \_\_p)
- `_Node_alloc_traits::pointer` **\_M\_get\_node** ()
- `size_t` **\_M\_get\_size** () const
- void **\_M\_inc\_size** (size\_t \_\_n)
- void **\_M\_put\_node** (\_Node\_ptr \_\_p) noexcept
- void **\_M\_set\_size** (size\_t \_\_n)

**Protected Attributes**

- `_List_impl` **\_M\_impl**

**5.103.1 Detailed Description**

`template<typename _Tp, typename _Alloc>`  
**class** `std::_List_base< _Tp, _Alloc >`

See `bits/stl_deque.h`'s `_Deque_base` for an explanation.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

**5.104 `std::_List_const_iterator< _Tp >` Struct Template Reference**

`#include <stl_list.h>`

**Public Types**

- typedef const `_List_node< _Tp >` **\_Node**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_List_iterator< _Tp >` **iterator**
- typedef `std::bidirectional_iterator_tag` **iterator\_category**
- typedef const `_Tp` \* **pointer**
- typedef const `_Tp` & **reference**
- typedef `_Tp` **value\_type**

### Public Member Functions

- `_List_const_iterator` (const `__detail::_List_node_base *__x`) noexcept
- `_List_const_iterator` (const `iterator` &\_\_x) noexcept
- `iterator _M_const_cast` () const noexcept
- reference `operator*` () const noexcept
- `_List_const_iterator` & `operator++` () noexcept
- `_List_const_iterator` `operator++` (int) noexcept
- `_List_const_iterator` & `operator--` () noexcept
- `_List_const_iterator` `operator--` (int) noexcept
- pointer `operator->` () const noexcept

### Public Attributes

- const `__detail::_List_node_base * _M_node`

### Friends

- bool `operator==` (const `_List_const_iterator` &\_\_x, const `_List_const_iterator` &\_\_y) noexcept

#### 5.104.1 Detailed Description

```
template<typename _Tp>
struct std::_List_const_iterator< _Tp >
```

A list::const\_iterator.

All the functions are op overloads.

The documentation for this struct was generated from the following files:

- [stl\\_iterator\\_base\\_funcs.h](#)
- [stl\\_list.h](#)

### 5.105 std::\_List\_iterator< \_Tp > Struct Template Reference

```
#include <stl_list.h>
```

#### Public Types

- typedef `_List_node`< \_Tp > `_Node`
- typedef ptrdiff\_t `difference_type`
- typedef `std::bidirectional_iterator_tag` `iterator_category`
- typedef `_Tp *` `pointer`
- typedef `_Tp &` `reference`
- typedef `_Tp` `value_type`

#### Public Member Functions

- `_List_iterator` (`__detail::_List_node_base *__x`) noexcept
- `_List_iterator` `_M_const_cast` () const noexcept
- reference `operator*` () const noexcept
- `_List_iterator` & `operator++` () noexcept
- `_List_iterator` `operator++` (int) noexcept
- `_List_iterator` & `operator--` () noexcept
- `_List_iterator` `operator--` (int) noexcept
- pointer `operator->` () const noexcept

**Public Attributes**

- [\\_\\_detail::\\_List\\_node\\_base](#) \* **\_M\_node**

**Friends**

- bool **operator==** (const [\\_List\\_iterator](#) &\_\_x, const [\\_List\\_iterator](#) &\_\_y) noexcept

**5.105.1 Detailed Description**

```
template<typename _Tp>
struct std::_List_iterator<_Tp>
```

A list::iterator.

All the functions are op overloads.

The documentation for this struct was generated from the following files:

- [stl\\_iterator\\_base\\_funcs.h](#)
- [stl\\_list.h](#)

**5.106 std::\_List\_node<\_Tp> Struct Template Reference**

```
#include <stl_list.h>
```

Inheritance diagram for std::\_List\_node<\_Tp>:

**Public Types**

- typedef `_List_node_base` \* **\_Base\_ptr**
- typedef [\\_List\\_node](#) \* **\_Node\_ptr**

**Public Member Functions**

- `_List_node_base` \* **\_M\_base** ()
- const `_List_node_base` \* **\_M\_base** () const
- void **\_M\_hook** (`_List_node_base` \*const \_\_position) noexcept
- [\\_Node\\_ptr](#) **\_M\_node\_ptr** ()
- void **\_M\_reverse** () noexcept
- void **\_M\_transfer** (`_List_node_base` \*const \_\_first, `_List_node_base` \*const \_\_last) noexcept
- void **\_M\_unhook** () noexcept



- `_Tp * _M_valptr ()`
- `_Tp const * _M_valptr () const`

#### Static Public Member Functions

- static void **swap** (`_List_node_base &__x`, `_List_node_base &__y`) noexcept

#### Public Attributes

- `_List_node_base * _M_next`
- `_List_node_base * _M_prev`
- `__gnu_cxx::__aligned_membuf<_Tp> _M_storage`

#### 5.106.1 Detailed Description

**template<typename \_Tp>**  
**struct std::\_List\_node<\_Tp>**

An actual node in the list.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

#### 5.107 std::\_\_detail::\_List\_node\_base Struct Reference

```
#include <stl_list.h>
```

Inheritance diagram for `std::__detail::_List_node_base`:



#### Public Types

- typedef `_List_node_base * _Base_ptr`

#### Public Member Functions

- `_List_node_base * _M_base ()`
- `const _List_node_base * _M_base () const`
- `void _M_hook (_List_node_base *const __position)` noexcept

- void **\_M\_reverse** () noexcept
- void **\_M\_transfer** ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) noexcept
- void **\_M\_unhook** () noexcept

### Static Public Member Functions

- static void **swap** ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

### Public Attributes

- [\\_List\\_node\\_base](#) \* **\_M\_next**
- [\\_List\\_node\\_base](#) \* **\_M\_prev**

#### 5.107.1 Detailed Description

Common part of a node in the list.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.108 std::\_\_detail::\_List\_node\_header Struct Reference

```
#include <stl_list.h>
```

Inheritance diagram for std::\_\_detail::\_List\_node\_header:



### Public Types

- typedef [\\_List\\_node\\_base](#) \* **\_Base\_ptr**

### Public Member Functions

- **\_List\_node\_header** ([\\_List\\_node\\_header](#) &&\_\_x) noexcept
- [\\_List\\_node\\_base](#) \* **\_M\_base** ()
- const [\\_List\\_node\\_base](#) \* **\_M\_base** () const
- void **\_M\_hook** ([\\_List\\_node\\_base](#) \*const \_\_position) noexcept
- void **\_M\_init** () noexcept
- void **\_M\_move\_nodes** ([\\_List\\_node\\_header](#) &&\_\_x)

- void **\_M\_reverse** () noexcept
- void **\_M\_transfer** ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) noexcept
- void **\_M\_unhook** () noexcept

#### Static Public Member Functions

- static void **swap** ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

#### Public Attributes

- [\\_List\\_node\\_base](#) \* **\_M\_next**
- [\\_List\\_node\\_base](#) \* **\_M\_prev**
- [size\\_t](#) **\_M\_size**

#### 5.108.1 Detailed Description

The list node header.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

### 5.109 `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser` Struct Reference

```
#include <losertree.h>
```

#### Public Attributes

- [\\_Tp](#) **\_M\_key**
- [int](#) **\_M\_source**
- [bool](#) **\_M\_sup**

#### 5.109.1 Detailed Description

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser
```

Internal representation of a `_LoserTree` element.

#### 5.109.2 Member Data Documentation

##### `_M_key`

```
template<typename _Tp, typename _Compare>
_Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key
_M_key of the element in the _LoserTree.
```

##### `_M_source`

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source
__index of the __source __sequence.
```

**\_M\_sup**

```
template<typename _Tp, typename _Compare>
bool __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_sup
flag, true iff this is a "maximum" __sentinel.
```

The documentation for this struct was generated from the following file:

- [losertree.h](#)

**5.110 `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>::_Loser` Struct Reference**

```
#include <losertree.h>
```

**Public Attributes**

- const \_Tp \* **\_M\_keyp**
- int **\_M\_source**
- bool **\_M\_sup**

**5.110.1 Detailed Description**

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>::_Loser
```

Internal representation of `_LoserTree` \_\_elements.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

**5.111 `__gnu_parallel::_LoserTree<__stable, _Tp, _Compare>` Class Template Reference**

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTree<__stable, _Tp, _Compare>`:

**Public Member Functions**

- **\_LoserTree** (unsigned int \_\_k, \_Compare \_\_comp)

- void [\\_\\_delete\\_min\\_insert](#) (\_Tp \_\_key, bool \_\_sup)
- int [\\_\\_get\\_min\\_source](#) ()
- void [\\_\\_init](#) ()
- unsigned int [\\_\\_init\\_winner](#) (unsigned int \_\_root)
- void [\\_\\_insert\\_start](#) (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

### Protected Attributes

- unsigned int [\\_M\\_ik](#)
- unsigned int [\\_M\\_log\\_k](#)
- unsigned int [\\_M\\_offset](#)

### 5.111.1 Detailed Description

template<bool \_\_stable, typename \_Tp, typename \_Compare>  
class [\\_\\_gnu\\_parallel::LoserTree](#)< \_\_stable, \_Tp, \_Compare >

Stable [\\_LoserTree](#) variant.

Provides the stable implementations of [insert\\_start](#), [\\_\\_init\\_winner](#), [\\_\\_init](#) and [\\_\\_delete\\_min\\_insert](#).

Unstable variant is done using partial specialisation below.

### 5.111.2 Member Function Documentation

#### [\\_\\_delete\\_min\\_insert\(\)](#)

```
template<bool __stable, typename _Tp, typename _Compare>
void \_\_gnu\_parallel::LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (
    _Tp __key,
    bool __sup) [inline]
```

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

#### [\\_\\_get\\_min\\_source\(\)](#)

```
template<typename _Tp, typename _Compare>
int \_\_gnu\_parallel::LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline], [inherited]
```

#### Returns

the index of the sequence with the smallest element.

References [\\_M\\_losers](#).

#### [\\_\\_insert\\_start\(\)](#)

```
template<typename _Tp, typename _Compare>
void \_\_gnu\_parallel::LoserTreeBase< _Tp, _Compare >::__insert_start (
    const _Tp & __key,
    int __source,
    bool __sup) [inline], [inherited]
```

Initializes the sequence "[\\_M\\_source](#)" with the element "[\\_\\_key](#)".

#### Parameters

|                          |                                                                                    |
|--------------------------|------------------------------------------------------------------------------------|
| <a href="#">__key</a>    | the element to insert                                                              |
| <a href="#">__source</a> | <a href="#">__index</a> of the <a href="#">__source</a> <a href="#">__sequence</a> |

|                    |                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------|
| <code>__sup</code> | flag that determines whether the value to insert is an explicit <code>__supremum</code> . |
|--------------------|-------------------------------------------------------------------------------------------|

References [\\_M\\_first\\_insert](#), and [\\_M\\_losers](#).

### 5.111.3 Member Data Documentation

#### `_M_log_k`

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected], [inherited]
log_2(_M_k)
```

Referenced by [\\_LoserTreeBase\(\)](#).

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.112 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- void [\\_\\_delete\\_min\\_insert](#) (`_Tp` `__key`, bool `__sup`)
- void [\\_\\_delete\\_min\\_insert](#) (`_Tp` `__key`, bool `__sup`)
- int [\\_\\_get\\_min\\_source](#) ()
- void `__init` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- unsigned int [\\_\\_init\\_winner](#) (unsigned int `__root`)
- void [\\_\\_insert\\_start](#) (const `_Tp` & `__key`, int `__source`, bool `__sup`)

## Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_log_k`
- unsigned int `_M_offset`

### 5.112.1 Detailed Description

`template<typename _Tp, typename _Compare>`  
`class __gnu_parallel::_LoserTree< false, _Tp, _Compare >`

Unstable `_LoserTree` variant.

Stability (non-stable here) is selected with partial specialization.

### 5.112.2 Member Function Documentation

#### `__delete_min_insert()` [1/2]

```
void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (
    _Tp __key,
    bool __sup) [inline]
```

Delete the smallest element and insert a new element from the previously smallest element's sequence.  
 This implementation is stable.

#### `__delete_min_insert()` [2/2]

```
template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__delete_min_insert (
    _Tp __key,
    bool __sup) [inline]
```

Delete the `_M_key` smallest element and insert the element `__key` instead.

#### Parameters

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>__key</code> | the <code>_M_key</code> to insert                            |
| <code>__sup</code> | true iff <code>__key</code> is an explicitly marked supremum |

#### `__get_min_source()`

```
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline]
```

#### Returns

the index of the sequence with the smallest element.

#### `__init_winner()`

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__init_winner (
    unsigned int __root) [inline]
```

Computes the winner of the competition at position "`__root`".

Called recursively (starting at 0) to build the initial tree.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__root</code> | __index of the "game" to start. |
|---------------------|---------------------------------|

References [\\_\\_init\\_winner\(\)](#).

Referenced by [\\_\\_init\\_winner\(\)](#).

**`__insert_start()`**

```
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
    const _Tp & __key,
    int __source,
    bool __sup) [inline]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

## Parameters

|                       |                                                                             |
|-----------------------|-----------------------------------------------------------------------------|
| <code>__key</code>    | the element to insert                                                       |
| <code>__source</code> | __index of the <code>__source</code> __sequence                             |
| <code>__sup</code>    | flag that determines whether the value to insert is an explicit __supremum. |

**5.112.3 Member Data Documentation****`_M_log_k`**

```
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected]
log_2{_M_k}
```

The documentation for this class was generated from the following file:

- [losertree.h](#)

**5.113 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference**

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`:

**Classes**

- struct [\\_Loser](#)



## Public Member Functions

- [\\_LoserTreeBase](#) (unsigned int \_\_k, \_Compare \_\_comp)
- [~\\_LoserTreeBase](#) ()
- int [\\_\\_get\\_min\\_source](#) ()
- void [\\_\\_insert\\_start](#) (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- [\\_Compare](#) [\\_M\\_comp](#)
- bool [\\_M\\_first\\_insert](#)
- unsigned int [\\_M\\_ik](#)
- unsigned int [\\_M\\_k](#)
- unsigned int [\\_M\\_log\\_k](#)
- [\\_Loser](#) \* [\\_M\\_losers](#)
- unsigned int [\\_M\\_offset](#)

### 5.113.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >
```

Guarded loser/tournament tree.

The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine.

This is a well-performing variant.

#### Parameters

|                       |                                                                      |
|-----------------------|----------------------------------------------------------------------|
| <code>_Tp</code>      | the element type                                                     |
| <code>_Compare</code> | the comparator to use, defaults to <code>std::less&lt;_Tp&gt;</code> |

### 5.113.2 Constructor & Destructor Documentation

#### `_LoserTreeBase()`

```
template<typename _Tp, typename _Compare>
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase (
    unsigned int __k,
    _Compare __comp) [inline]
```

The constructor.

#### Parameters

|                     |                                   |
|---------------------|-----------------------------------|
| <code>__k</code>    | The number of sequences to merge. |
| <code>__comp</code> | The comparator to use.            |

References [\\_\\_gnu\\_parallel::\\_\\_rd\\_log2\(\)](#), [\\_M\\_comp](#), [\\_M\\_first\\_insert](#), [\\_M\\_log\\_k](#), and [\\_M\\_losers](#).

#### `~_LoserTreeBase()`

```
template<typename _Tp, typename _Compare>
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase () [inline]
```

The destructor.

References [\\_M\\_losers](#).

### 5.113.3 Member Function Documentation

#### `__get_min_source()`

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__get_min_source () [inline]
```

#### Returns

the index of the sequence with the smallest element.

References [\\_M\\_losers](#).

#### `__insert_start()`

```
template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__insert_start (
    const _Tp & __key,
    int __source,
    bool __sup) [inline]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

#### Parameters

|                       |                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| <code>__key</code>    | the element to insert                                                                     |
| <code>__source</code> | <code>__index</code> of the <code>__source</code> sequence                                |
| <code>__sup</code>    | flag that determines whether the value to insert is an explicit <code>__supremum</code> . |

References [\\_M\\_first\\_insert](#), and [\\_M\\_losers](#).

### 5.113.4 Member Data Documentation

#### `_M_comp`

```
template<typename _Tp, typename _Compare>
_Compare __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_M_comp [protected]
_Compare to use.
Referenced by \_LoserTreeBase\(\).
```

#### `_M_first_insert`

```
template<typename _Tp, typename _Compare>
bool __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_M_first_insert [protected]
State flag that determines whether the _LoserTree is empty.
Only used for building the _LoserTree.
Referenced by \_LoserTreeBase\(\), and \_\_insert\_start\(\).
```

#### `_M_log_k`

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_M_log_k [protected]
log_2{_M_k}
Referenced by \_LoserTreeBase\(\).
```

**\_M\_losers**

```
template<typename _Tp, typename _Compare>
```

```
__Loser* __gnu_parallel::__LoserTreeBase< _Tp, _Compare >::_M_losers [protected]
```

```
__LoserTree __elements.
```

Referenced by [\\_\\_LoserTreeBase\(\)](#), [~\\_\\_LoserTreeBase\(\)](#), [\\_\\_get\\_min\\_source\(\)](#), and [\\_\\_insert\\_start\(\)](#).

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.114 [\\_\\_gnu\\_parallel::\\_\\_LoserTreePointer< \\_\\_stable, \\_Tp, \\_Compare >](#) Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for [\\_\\_gnu\\_parallel::\\_\\_LoserTreePointer< \\_\\_stable, \\_Tp, \\_Compare >](#):

**Public Member Functions**

- [\\_\\_LoserTreePointer](#) (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void [\\_\\_delete\\_min\\_insert](#) (const \_Tp &\_\_key, bool \_\_sup)
- int [\\_\\_get\\_min\\_source](#) ()
- void [\\_\\_init](#) ()
- unsigned int [\\_\\_init\\_winner](#) (unsigned int \_\_root)
- void [\\_\\_insert\\_start](#) (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

**Protected Attributes**

- unsigned int [\\_M\\_ik](#)
- unsigned int [\\_M\\_offset](#)

**5.114.1 Detailed Description**

```
template<bool __stable, typename _Tp, typename _Compare>
```

```
class __gnu_parallel::__LoserTreePointer< __stable, _Tp, _Compare >
```

Stable `__LoserTree` implementation.

The unstable variant is implemented using partial instantiation below.  
The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.115 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTreePointer` (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- `_LoserTreePointer` (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- `void __delete_min_insert` (const \_Tp &\_\_key, bool \_\_sup)
- `void __delete_min_insert` (const \_Tp &\_\_key, bool \_\_sup)
- `int __get_min_source` ()
- `void __init` ()
- `void __init` ()
- `unsigned int __init_winner` (unsigned int \_\_root)
- `unsigned int __init_winner` (unsigned int \_\_root)
- `void __insert_start` (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

### Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_offset`

#### 5.115.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >
```

Unstable `_LoserTree` implementation.

The stable variant is above.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.116 `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>`:



#### Classes

- struct [\\_Loser](#)

#### Public Member Functions

- `_LoserTreePointerBase` (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)<\_Tp>())
- int `__get_min_source` ()
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

#### Protected Attributes

- \_Compare `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- [\\_Loser](#) \* `_M_losers`
- unsigned int `_M_offset`

#### 5.116.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>
```

Base class of `_Loser` Tree implementation using pointers.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.117 `__gnu_parallel::_LoserTreePointerUnguarded<__stable, _Tp, _Compare>` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



### Public Member Functions

- **\_LoserTreePointerUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **\_\_delete\_min\_insert** (const \_Tp &\_\_key, bool \_\_sup)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

#### 5.117.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >
```

Stable unguarded \_LoserTree variant storing pointers.

Unstable variant is implemented below using partial specialization.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.118 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



### Public Member Functions

- **\_LoserTreePointerUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- **\_LoserTreePointerUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **\_\_delete\_min\_insert** (const \_Tp &\_\_key, bool \_\_sup)
- void **\_\_delete\_min\_insert** (const \_Tp &\_\_key, bool \_\_sup)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

#### 5.118.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >
```

Unstable unguarded \_LoserTree variant storing pointers.

Stable variant is above.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.119 `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>`:



### Public Member Functions

- `_LoserTreePointerUnguardedBase` (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less<\\_Tp>](#)())
- int `__get_min_source` ()
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

#### 5.119.1 Detailed Description

```
template<typename _Tp, typename _Compare>
```

```
class __gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>
```

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.120 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference

```
#include <multiway_merge.h>
```

### Static Public Attributes

- static const bool `_M_use_pointer`



### 5.120.1 Detailed Description

```
template<typename _Tp>
struct __gnu_parallel::_LoserTreeTraits< _Tp >
```

Traits for determining whether the loser tree should use pointers or copies.

The field "`_M_use_pointer`" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };
template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };
```

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>_Tp</code> | type to give the loser tree traits for. |
|------------------|-----------------------------------------|

### 5.120.2 Member Data Documentation

#### `_M_use_pointer`

```
template<typename _Tp>
const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer [static]
```

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 5.121 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`:



**Public Member Functions**

- **`_LoserTreeUnguarded`** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **`__delete_min_insert`** (\_Tp \_\_key, bool)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int \_\_root)
- void **`__insert_start`** (const \_Tp &\_\_key, int \_\_source, bool)

**Protected Attributes**

- unsigned int **`_M_ik`**
- unsigned int **`_M_offset`**

**5.121.1 Detailed Description**

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
```

Stable implementation of unguarded `_LoserTree`.

Unstable variant is selected below with partial specialization.

The documentation for this class was generated from the following file:

- [losertree.h](#)

**5.122 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference**

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`:

**Public Member Functions**

- **`_LoserTreeUnguarded`** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- **`_LoserTreeUnguarded`** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **`__delete_min_insert`** (\_Tp \_\_key, bool)

- void **\_\_delete\_min\_insert** (\_Tp \_\_key, bool)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

#### 5.122.1 Detailed Description

template<typename \_Tp, typename \_Compare>  
class **\_\_gnu\_parallel::\_LoserTreeUnguarded**< false, \_Tp, \_Compare >

Non-Stable implementation of ungarded \_LoserTree.

Stable implementation is above.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 5.123 \_\_gnu\_parallel::\_LoserTreeUnguardedBase< \_Tp, \_Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_LoserTreeUnguardedBase< \_Tp, \_Compare >:



### Public Member Functions

- **\_LoserTreeUnguardedBase** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- int **\_\_get\_min\_source** ()
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

#### 5.123.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >
```

Base class for unguarded `_LoserTree` implementation.

The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.124 `__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >` Struct Template Reference

```
#include <pat_trie_base.hpp>
```

#### Public Types

- typedef `_Alloc` `allocator_type`
- typedef [detail::rebind\\_traits](#)`< _Alloc, Metadata >::const_reference` `const_reference`
- typedef `Metadata` `metadata_type`

#### Public Member Functions

- `const_reference` `get_metadata` () const

#### Public Attributes

- `metadata_type` `m_metadata`

#### 5.124.1 Detailed Description

```
template<typename Metadata, typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >
```

Metadata base primary template.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

### 5.125 `__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >` Struct Template Reference

```
#include <pat_trie_base.hpp>
```

## Public Types

- typedef `_Alloc` `allocator_type`
- typedef `_Alloc` `allocator_type`
- typedef `detail::rebind_traits<_Alloc, null_type>::const_reference` `const_reference`
- typedef `null_type` `metadata_type`
- typedef `null_type` `metadata_type`

## Public Member Functions

- `const_reference` `get_metadata` () const

## Public Attributes

- `metadata_type` `m_metadata`

### 5.125.1 Detailed Description

```
template<typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >
```

Specialization for null metadata.

The documentation for this struct was generated from the following file:

- `pat_trie_base.hpp`

## 5.126 \_\_gnu\_parallel::\_Multiplies<\_Tp1, \_Tp2, \_Result > Struct Template Reference

```
#include <base.h>
```

Inheritance diagram for `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result >`:



## Public Types

- typedef `_Arg1` `first_argument_type`
- typedef `_Result` `result_type`
- typedef `_Arg2` `second_argument_type`

**Public Member Functions**

- `_Result operator() (const _Tp1 &__x, const _Tp2 &__y) const`

**5.126.1 Detailed Description**

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__((*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0)))>
struct __gnu_parallel::__Multiplies<_Tp1, _Tp2, _Result>
```

Similar to std::multiplies, but allows two different types.

**5.126.2 Member Typedef Documentation****first\_argument\_type**

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function<_Arg1, _Arg2, _Result>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

**result\_type**

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function<_Arg1, _Arg2, _Result>::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function<_Arg1, _Arg2, _Result>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [base.h](#)

**5.127 std::\_\_fwdlst::\_\_Node<\_ValPtr> Struct Template Reference**

```
#include <forward_list.h>
```

Inheritance diagram for `std::__fwlist::_Node<_ValPtr>`:



### Public Types

- using `_Base_ptr`
- using `_Node_ptr`
- using `value_type`

### Public Member Functions

- `_Node` (`_Node` &&)=delete
- `_Base_ptr` `_M_base_ptr` () const
- `_Node_ptr` `_M_node_ptr` ()
- void `_M_reverse_after` () noexcept
- `_Base_ptr` `_M_transfer_after` (`_Base_ptr` \_\_begin, `_Base_ptr` \_\_end) noexcept
- const `value_type` \* `_M_valptr` () const noexcept
- `value_type` \* `_M_valptr` () noexcept

### Public Attributes

- `_Base_ptr` `_M_next`
- `_Uninit_storage` `_M_u`

#### 5.127.1 Detailed Description

`template<typename _ValPtr>`  
`struct std::__fwlist::_Node<_ValPtr>`

A helper node class for `forward_list`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.128 `__gnu_pbds::detail::pat_trie_base::_Node_base<_ATraits, Metadata>` Struct Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_base<_ATraits, Metadata>`:



### Public Types

- typedef `_ATraits::const_iterator` **a\_const\_iterator**
- typedef `detail::rebind_traits<_Alloc, _ATraits>::const_pointer` **a\_const\_pointer**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `detail::rebind_traits<_Alloc, _Node_base>::pointer` **node\_pointer**
- typedef `_ATraits::type_traits` **type\_traits**

### Public Member Functions

- `_Node_base` (`node_type` type)

### Public Attributes

- `node_pointer` **m\_p\_parent**
- `const node_type` **m\_type**

#### 5.128.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
```

```
struct __gnu_pbds::detail::pat_trie_base::_Node_base<_ATraits, Metadata>
```

Node base.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)



## 5.129 `std::__fwlist::__Node_base<_VoidPtr>` Struct Template Reference

```
#include <forward_list.h>
```

### Public Types

- `using _Base_ptr`

### Public Member Functions

- `_Node_base` (`_Node_base` &&\_\_x) noexcept
- `_Node_base` (const `_Node_base` &)=delete
- `_Base_ptr _M_base_ptr` () const
- `void _M_reverse_after` () noexcept
- `_Base_ptr _M_transfer_after` (`_Base_ptr` \_\_begin, `_Base_ptr` \_\_end) noexcept
- `_Node_base` & `operator=` (`_Node_base` &&\_\_x) noexcept
- `_Node_base` & `operator=` (const `_Node_base` &)=delete

### Public Attributes

- `_Base_ptr _M_next`

#### 5.129.1 Detailed Description

```
template<typename _VoidPtr>
struct std::__fwlist::__Node_base<_VoidPtr>
```

The node-base type for allocators that use fancy pointers.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.130 `__gnu_pbds::detail::pat_trie_base::_Node_citer<Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc>` Class Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_citer<Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc>`:



## Public Types

- typedef value\_type **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits< \\_Alloc, metadata\\_type >::const\\_reference](#) **metadata\_const\_reference**
- typedef Node::metadata\_type **metadata\_type**
- typedef value\_type **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef \_CIterator **value\_type**

## Public Member Functions

- **\_Node\_citer** (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- **\_Node\_citer** [get\\_child](#) (size\_type i) const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- size\_type [num\\_children](#) () const
- bool [operator!=](#) (const **\_Node\_citer** &other) const
- const\_reference [operator\\*](#) () const
- bool [operator==](#) (const **\_Node\_citer** &other) const
- [std::pair< a\\_const\\_iterator, a\\_const\\_iterator >](#) [valid\\_prefix](#) () const

## Public Attributes

- node\_pointer **m\_p\_nd**
- a\_const\_pointer **m\_p\_traits**

## Protected Types

- typedef Node::a\_const\_iterator **a\_const\_iterator**
- typedef Node::a\_const\_pointer **a\_const\_pointer**
- typedef [rebind\\_traits< \\_Alloc, Inode >::const\\_pointer](#) **inode\_const\_pointer**
- typedef [rebind\\_traits< \\_Alloc, Inode >::pointer](#) **inode\_pointer**
- typedef [rebind\\_traits< \\_Alloc, Leaf >::const\\_pointer](#) **leaf\_const\_pointer**
- typedef [rebind\\_traits< \\_Alloc, Leaf >::pointer](#) **leaf\_pointer**
- typedef [rebind\\_traits< \\_Alloc, Node >::pointer](#) **node\_pointer**

### 5.130.1 Detailed Description

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>`

`class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`

Node const iterator.

### 5.130.2 Member Typedef Documentation

#### **metadata\_const\_reference**

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>`

typedef [rebind\\_traits<\\_Alloc,metadata\\_type>::const\\_reference](#) `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference`

Const metadata reference type.

## metadata\_type

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type
Metadata type.
```

### 5.130.3 Member Function Documentation

#### get\_child()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
__Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::get_child (
    size_type i) const [inline]
```

Returns a `__const` node `__iterator` to the corresponding node's *i*-th child.

#### get\_metadata()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata () const [inline]
```

Metadata access.

#### num\_children()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children () const [inline]
```

Returns the number of children in the corresponding node.

Referenced by `__gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::operator*()`.

#### operator"!="()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!= (
    const __Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline]
```

Compares content (negatively) to a different iterator object.

#### operator\*()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _C
Iterator, Iterator, _Alloc >::operator* () const [inline]
```

Const access; returns the `__const` iterator\* associated with the current leaf.

### `operator==()`

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator== (
    const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline]
```

Compares content to a different iterator object.

### `valid_prefix()`

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
std::pair< a_const_iterator, a_const_iterator > __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const [inline]
```

Subtree valid prefix.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.131 `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >` Class Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`:



### Public Types

- typedef value\_type **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits< \\_Alloc, metadata\\_type >::const\\_reference](#) **metadata\_const\_reference**
- typedef `Node::metadata_type` **metadata\_type**
- typedef value\_type **reference**
- typedef `base_type::size_type` **size\_type**
- typedef Iterator **value\_type**

## Public Member Functions

- `_Node_iter` (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- `_Node_iter get_child` (size\_type i) const
- `metadata_const_reference get_metadata` () const
- size\_type `num_children` () const
- bool `operator!=` (const `_Node_citer` &other) const
- reference `operator*` () const
- bool `operator==` (const `_Node_citer` &other) const
- `std::pair`< a\_const\_iterator, a\_const\_iterator > `valid_prefix` () const

## Public Attributes

- node\_pointer `m_p_nd`
- a\_const\_pointer `m_p_traits`

## Protected Types

- typedef Node::a\_const\_iterator `a_const_iterator`
- typedef `rebind_traits`< \_Alloc, Inode >::const\_pointer `inode_const_pointer`
- typedef `rebind_traits`< \_Alloc, Leaf >::const\_pointer `leaf_const_pointer`
- typedef `rebind_traits`< \_Alloc, Leaf >::pointer `leaf_pointer`

### 5.131.1 Detailed Description

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>`  
`class __gnu_pbds::detail::pat_trie_base::_Node_iter`< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >

Node iterator.

### 5.131.2 Member Typedef Documentation

#### `metadata_const_reference`

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference [inherited]
Const metadata reference type.
```

#### `metadata_type`

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type [inherited]
Metadata type.
```

### 5.131.3 Member Function Documentation

#### `get_child()`

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
_Node_iter __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
```

```
Iterator, _Alloc >::get_child (
    size_type i) const [inline]
```

Returns a node `__iterator` to the corresponding node's i-th child.

### **get\_metadata()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata () const [inline], [inherited]
```

Metadata access.

### **num\_children()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children () const [inline], [inherited]
```

Returns the number of children in the corresponding node.  
Referenced by `__gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::operator*()`.

### **operator"!=()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!= (
    const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

### **operator\*()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator* () const [inline]
```

Access; returns the iterator\* associated with the current leaf.

### **operator==(())**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator== (
    const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline], [inherited]
```

Compares content to a different iterator object.

### **valid\_prefix()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
std::pair< a_const_iterator, a_const_iterator > __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const [inline], [inherited]
```

Subtree valid prefix.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.132 \_\_gnu\_debug::Not\_equal\_to<\_Type> Class Template Reference

```
#include <safe_sequence.h>
```

### Public Member Functions

- **Not\_equal\_to** (const \_Type &\_\_v)
- bool **operator()** (const \_Type &\_\_x) const

#### 5.132.1 Detailed Description

```
template<typename _Type>
class __gnu_debug::Not_equal_to<_Type>
```

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both bind1st and not\_equal.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.133 std::Not\_fn<\_Fn> Class Template Reference

```
#include <functional>
```

### Public Member Functions

- template<typename \_Fn2>  
constexpr **Not\_fn** (\_Fn2 &&\_\_fn, int)
- **Not\_fn** ([Not\\_fn](#) &&\_\_fn)=default
- **Not\_fn** (const [Not\\_fn](#) &\_\_fn)=default
- template<typename... \_Args, typename = enable\_if\_t<![is\\_invocable](#)<\_Fn &&, \_Args...>::value>>  
void **operator()** (\_Args &&... \_\_args) &&=delete
- template<typename... \_Args, typename = enable\_if\_t<![is\\_invocable](#)<\_Fn &&, \_Args...>::value>>  
constexpr decltype(\_S\_not<\_\_inv\_res\_t<\_Fn &&, \_Args...>>()) **operator()** (\_Args &&... \_\_args)  
&&noexcept([is\\_nothrow\\_invocable](#)<\_Fn &&, \_Args...>::value &&noexcept(\_S\_not<\_\_inv\_res\_t<\_Fn &&, \_Args...>>()))
- template<typename... \_Args, typename = enable\_if\_t<![is\\_invocable](#)<\_Fn &, \_Args...>::value>>  
void **operator()** (\_Args &&... \_\_args) &=delete
- template<typename... \_Args, typename = enable\_if\_t<![is\\_invocable](#)<\_Fn &, \_Args...>::value>>  
constexpr decltype(\_S\_not<\_\_inv\_res\_t<\_Fn &, \_Args...>>()) **operator()** (\_Args &&... \_\_args) &noexcept(↵  
\_S\_not<\_\_inv\_res\_t<\_Fn &, \_Args...>>::value &&noexcept(\_S\_not<\_\_inv\_res\_t<\_Fn &, \_Args...>>()))
- template<typename... \_Args, typename = enable\_if\_t<![is\\_invocable](#)<\_Fn const &&, \_Args...>::value>>  
void **operator()** (\_Args &&... \_\_args) const &&=delete
- template<typename... \_Args, typename = enable\_if\_t<![is\\_invocable](#)<\_Fn const &&, \_Args...>::value>>  
constexpr decltype(\_S\_not<\_\_inv\_res\_t<\_Fn const &&, \_Args...>>()) **operator()** (\_Args &&... \_\_args) const  
&&noexcept([is\\_nothrow\\_invocable](#)<\_Fn const &&, \_Args...>::value &&noexcept(\_S\_not<\_\_inv\_res\_t<\_Fn const &&, \_Args...>>()))
- template<typename... \_Args, typename = enable\_if\_t<![is\\_invocable](#)<\_Fn const &, \_Args...>::value>>  
void **operator()** (\_Args &&... \_\_args) const &=delete

- `template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn const &, _Args...>::value>>`  
`constexpr decltype(_S_not< __inv_res_t< _Fn const &, _Args... >>()) operator() (_Args &&... __args) const`  
`&noexcept(__is_nothrow_invocable< _Fn const &, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn`  
`const &, _Args... >>()))`

#### 5.133.1 Detailed Description

`template<typename _Fn>`  
`class std::__Not_fn< _Fn >`

Generalized negator.

The documentation for this class was generated from the following file:

- [functional](#)

### 5.134 `__gnu_parallel::__Nothing` Struct Reference

`#include <for_each_selectors.h>`

#### Public Member Functions

- `template<typename _It>`  
`void operator() (_It __i)`

#### 5.134.1 Detailed Description

Functor doing nothing.

For some `__reduction` tasks (this is not a function object, but is passed as `__selector __dummy` parameter.

#### 5.134.2 Member Function Documentation

##### `operator()`

`template<typename _It>`  
`void __gnu_parallel::__Nothing::operator() (`  
`_It __i) [inline]`

Functor execution.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>↩</code>   | iterator referencing object. |
| <code>__↩</code> |                              |
| <code>↩</code>   |                              |
| <code>__↩</code> |                              |
| <code>i</code>   |                              |

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.135 `__gnu_parallel::__Piece< _DifferenceTp >` Struct Template Reference

`#include <multiway_mergesort.h>`

#### Public Types

- `typedef _DifferenceTp _DifferenceType`



## Public Attributes

- `_DifferenceType` [\\_M\\_begin](#)
- `_DifferenceType` [\\_M\\_end](#)

### 5.135.1 Detailed Description

```
template<typename _DifferenceTp>
struct __gnu_parallel::_Piece< _DifferenceTp >
```

Subsequence description.

### 5.135.2 Member Data Documentation

#### `_M_begin`

```
template<typename _DifferenceTp>
_DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_begin
```

Begin of subsequence.

#### `_M_end`

```
template<typename _DifferenceTp>
_DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end
```

End of subsequence.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 5.136 `std::_Placeholder< _Num >` Struct Template Reference

```
#include <functional>
```

### 5.136.1 Detailed Description

```
template<int _Num>
struct std::_Placeholder< _Num >
```

The type of placeholder objects defined by libstdc++.

Since

C++11

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.137 `__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >` Struct Template Reference

```
#include <base.h>
```

Inheritance diagram for `__gnu_parallel::__Plus<_Tp1, _Tp2, _Result>`:



### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- `_Result` **operator()** (const `_Tp1` &`_x`, const `_Tp2` &`_y`) const

#### 5.137.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__((*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0)))>
struct __gnu_parallel::__Plus<_Tp1, _Tp2, _Result>
```

Similar to `std::plus`, but allows two different types.

#### 5.137.2 Member Typedef Documentation

##### `first_argument_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary\_function<\_Arg1, \_Arg2, \_Result>::first\_argument\_type [inherited]
first\_argument\_type is the type of the first argument
```

##### `result_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary\_function<\_Arg1, \_Arg2, \_Result>::result\_type [inherited]
result\_type is the return type
```

##### `second_argument_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary\_function<\_Arg1, \_Arg2, \_Result>::second\_argument\_type [inherited]
```

`second_argument_type` is the type of the second argument  
 The documentation for this struct was generated from the following file:

- [base.h](#)

### 5.138 `__gnu_parallel::PMWMSSortingData<_RAIter>` Struct Template Reference

```
#include <multiway_mergesort.h>
```

#### Public Types

- typedef `_TraitsType::difference_type` **`_DifferenceType`**
- typedef `std::iterator_traits<_RAIter>` **`_TraitsType`**
- typedef `_TraitsType::value_type` **`_ValueType`**

#### Public Attributes

- `_ThreadIndex` `_M_num_threads`
- `_DifferenceType` \* `_M_offsets`
- `std::vector<_Piece<_DifferenceType>>` \* `_M_pieces`
- `_ValueType` \* `_M_samples`
- `_RAIter` `_M_source`
- `_DifferenceType` \* `_M_starts`
- `_ValueType` \*\* `_M_temporary`

#### 5.138.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::PMWMSSortingData<_RAIter>
```

Data accessed by all threads.  
 PMWMS = parallel multiway mergesort

#### 5.138.2 Member Data Documentation

##### `_M_num_threads`

```
template<typename _RAIter>
\_ThreadIndex \_\_gnu\_parallel::PMWMSSortingData<\_RAIter>::\_M\_num\_threads
Number of threads involved.
Referenced by \_\_gnu\_parallel::parallel\_sort\_mwms\(\), and \_\_gnu\_parallel::parallel\_sort\_mwms\_pu\(\).
```

##### `_M_offsets`

```
template<typename _RAIter>
\_DifferenceType* \_\_gnu\_parallel::PMWMSSortingData<\_RAIter>::\_M\_offsets
Offsets to add to the found positions.
Referenced by \_\_gnu\_parallel::parallel\_sort\_mwms\(\).
```

##### `_M_pieces`

```
template<typename _RAIter>
std::vector<\_Piece<\_DifferenceType>>* \_\_gnu\_parallel::PMWMSSortingData<\_RAIter>::\_M\_pieces
Pieces of data to merge [thread][__sequence].
Referenced by \_\_gnu\_parallel::parallel\_sort\_mwms\(\), and \_\_gnu\_parallel::parallel\_sort\_mwms\_pu\(\).
```

### `_M_samples`

```
template<typename _RAIter>
_ValueType* __gnu_parallel::_PMWSSortingData<_RAIter>::_M_samples
```

Samples.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_determine\\_samples\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#).

### `_M_source`

```
template<typename _RAIter>
_RAIter __gnu_parallel::_PMWSSortingData<_RAIter>::_M_source
```

Input `__begin`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_determine\\_samples\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_](#)

### `_M_starts`

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWSSortingData<_RAIter>::_M_starts
```

Start indices, per thread.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_determine\\_samples\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_](#)

### `_M_temporary`

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_PMWSSortingData<_RAIter>::_M_temporary
```

Storage in which to sort.

Referenced by [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#).

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 5.139 `__gnu_cxx::Pointer_adapter<_Storage_policy>` Class Template Reference

```
#include <pointer.h>
```

### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Storage_policy::element_type` **element\_type**
- typedef [std::random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef [\\_Pointer\\_adapter](#) **pointer**
- typedef `_Reference_type< element_type >::reference` **reference**
- typedef [\\_Unqualified\\_type](#)< `element_type` >::type **value\_type**

### Public Member Functions

- `template<typename _Up>`  
  **\_Pointer\_adapter** (`_Up *__arg`)
- **\_Pointer\_adapter** (`const _Pointer_adapter &__arg`)
- `template<typename _Up>`  
  **\_Pointer\_adapter** (`const _Pointer_adapter<_Up> &__arg`)
- **\_Pointer\_adapter** (`element_type *__arg=0`)
- **operator bool** () const
- `reference operator*` () const
- [\\_Pointer\\_adapter](#) & **operator++** ()

- `_Pointer_adapter operator++` (int)
- `_Pointer_adapter & operator+=` (int \_\_offset)
- `_Pointer_adapter & operator+=` (long \_\_offset)
- `_Pointer_adapter & operator+=` (long long \_\_offset)
- `_Pointer_adapter & operator+=` (short \_\_offset)
- `_Pointer_adapter & operator+=` (unsigned int \_\_offset)
- `_Pointer_adapter & operator+=` (unsigned long \_\_offset)
- `_Pointer_adapter & operator+=` (unsigned long long \_\_offset)
- `_Pointer_adapter & operator+=` (unsigned short \_\_offset)
- `template<typename _Up>`  
`std::ptrdiff_t operator-` (const `_Pointer_adapter<_Up>` & \_\_rhs) const
- `_Pointer_adapter & operator--` ()
- `_Pointer_adapter operator--` (int)
- `_Pointer_adapter & operator-=` (int \_\_offset)
- `_Pointer_adapter & operator-=` (long \_\_offset)
- `_Pointer_adapter & operator-=` (long long \_\_offset)
- `_Pointer_adapter & operator-=` (short \_\_offset)
- `_Pointer_adapter & operator-=` (unsigned int \_\_offset)
- `_Pointer_adapter & operator-=` (unsigned long \_\_offset)
- `_Pointer_adapter & operator-=` (unsigned long long \_\_offset)
- `_Pointer_adapter & operator-=` (unsigned short \_\_offset)
- `element_type * operator->` () const
- `template<typename _Up>`  
`_Pointer_adapter & operator=` (\_Up \* \_\_arg)
- `_Pointer_adapter & operator=` (const `_Pointer_adapter` & \_\_arg)
- `template<typename _Up>`  
`_Pointer_adapter & operator=` (const `_Pointer_adapter<_Up>` & \_\_arg)
- `reference operator[]` (std::ptrdiff\_t \_\_index) const

## Friends

- `_Pointer_adapter operator+` (const `_Pointer_adapter` & \_\_lhs, int \_\_offset)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & \_\_lhs, long \_\_offset)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & \_\_lhs, long long \_\_offset)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & \_\_lhs, short \_\_offset)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & \_\_lhs, unsigned int \_\_offset)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & \_\_lhs, unsigned long \_\_offset)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & \_\_lhs, unsigned long long \_\_offset)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` & \_\_lhs, unsigned short \_\_offset)
- `_Pointer_adapter operator+` (int \_\_offset, const `_Pointer_adapter` & \_\_rhs)
- `_Pointer_adapter operator+` (long \_\_offset, const `_Pointer_adapter` & \_\_rhs)
- `_Pointer_adapter operator+` (long long \_\_offset, const `_Pointer_adapter` & \_\_rhs)
- `_Pointer_adapter operator+` (short \_\_offset, const `_Pointer_adapter` & \_\_rhs)
- `_Pointer_adapter operator+` (unsigned int \_\_offset, const `_Pointer_adapter` & \_\_rhs)
- `_Pointer_adapter operator+` (unsigned long \_\_offset, const `_Pointer_adapter` & \_\_rhs)
- `_Pointer_adapter operator+` (unsigned long long \_\_offset, const `_Pointer_adapter` & \_\_rhs)
- `_Pointer_adapter operator+` (unsigned short \_\_offset, const `_Pointer_adapter` & \_\_rhs)
- `template<typename _Up>`  
`std::ptrdiff_t operator-` (\_Up \* \_\_lhs, const `_Pointer_adapter` & \_\_rhs)
- `template<typename _Up>`  
`std::ptrdiff_t operator-` (const `_Pointer_adapter` & \_\_lhs, \_Up \* \_\_rhs)

- `std::ptrdiff_t operator-` (const [\\_Pointer\\_adapter](#) &\_\_lhs, element\_type \*\_\_rhs)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, int \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, long \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, long long \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, short \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned int \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned long \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned long long \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned short \_\_offset)
- `std::ptrdiff_t operator-` (element\_type \*\_\_lhs, const [\\_Pointer\\_adapter](#) &\_\_rhs)
- `std::strong_ordering operator<=>` (const [\\_Pointer\\_adapter](#) &\_\_lhs, const [\\_Pointer\\_adapter](#) &\_\_rhs) noexcept

### 5.139.1 Detailed Description

```
template<typename _Storage_policy>
class __gnu_cxx::_Pointer_adapter< _Storage_policy >
```

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from `ext/cast.h`. This allows pointer cast operations to be overloaded as necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

```
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const
_Tp> >; _Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* const == const _Pointer_
adapter<_Std_pointer_impl<const _Tp> >;
```

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.140 `__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>` Class Template Reference

```
#include <base.h>
```

### Public Types

- typedef `_DifferenceTp` **`_DifferenceType`**
- typedef [\\_PseudoSequenceIterator](#)< `_Tp`, `uint64_t` > **`iterator`**

### Public Member Functions

- [\\_PseudoSequence](#) (const `_Tp` &\_\_val, `_DifferenceType` \_\_count)
- [iterator begin](#) () const
- [iterator end](#) () const

### 5.140.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >
```

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

#### Parameters

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>_Tp</code>           | Sequence <code>_M</code> value type. |
| <code>_DifferenceTp</code> | Sequence difference type.            |

### 5.140.2 Constructor & Destructor Documentation

#### `_PseudoSequence()`

```
template<typename _Tp, typename _DifferenceTp>
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::_PseudoSequence (
    const _Tp & __val,
    _DifferenceType __count) [inline]
```

Constructor.

#### Parameters

|                      |                             |
|----------------------|-----------------------------|
| <code>__val</code>   | Element of the sequence.    |
| <code>__count</code> | Number of (virtual) copies. |

### 5.140.3 Member Function Documentation

#### `begin()`

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::begin () const [inline]
```

Begin iterator.

#### `end()`

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::end () const [inline]
```

End iterator.

The documentation for this class was generated from the following file:

- [base.h](#)

## 5.141 `__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >` Class Template Reference

```
#include <base.h>
```

#### Public Types

- typedef `_DifferenceTp` `_DifferenceType`

**Public Member Functions**

- `_PseudoSequencerIterator` (const `_Tp` &\_\_val, `_DifferenceType` \_\_pos)
- `bool operator!=` (const `_PseudoSequencerIterator` &\_\_i2)
- `const _Tp & operator*` () const
- `_PseudoSequencerIterator & operator++` ()
- `_PseudoSequencerIterator operator++` (int)
- `_DifferenceType operator-` (const `_PseudoSequencerIterator` &\_\_i2)
- `bool operator==` (const `_PseudoSequencerIterator` &\_\_i2)
- `const _Tp & operator[]` (`_DifferenceType`) const

**5.141.1 Detailed Description**

`template<typename _Tp, typename _DifferenceTp>`  
**class** `__gnu_parallel::_PseudoSequencerIterator<_Tp, _DifferenceTp>`

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.

**Parameters**

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>_Tp</code>           | Sequence <code>_M_value</code> type. |
| <code>_DifferenceTp</code> | Sequence difference type.            |

The documentation for this class was generated from the following file:

- [base.h](#)

**5.142 `__gnu_parallel::__QSBThreadLocal<_RAIter>` Struct Template Reference**

```
#include <balanced_quicksort.h>
```

**Public Types**

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::pair<_RAIter, _RAIter> _Piece`
- `typedef std::iterator_traits<_RAIter> _TraitsType`

**Public Member Functions**

- `_QSBThreadLocal` (int \_\_queue\_size)

**Public Attributes**

- `volatile _DifferenceType * _M_elements_leftover`
- `_Piece _M_global`
- `_Piece _M_initial`
- `_RestrictedBoundedConcurrentQueue<_Piece> _M_leftover_parts`
- `_ThreadIndex _M_num_threads`

**5.142.1 Detailed Description**

`template<typename _RAIter>`  
**struct** `__gnu_parallel::__QSBThreadLocal<_RAIter>`

Information local to one thread in the parallel quicksort run.



### 5.142.2 Member Typedef Documentation

#### **`_Piece`**

```
template<typename _RAIter>
```

```
typedef std::pair<_RAIter, _RAIter> __gnu_parallel::__QSBThreadLocal< _RAIter >::_Piece
```

Continuous part of the sequence, described by an iterator pair.

### 5.142.3 Constructor & Destructor Documentation

#### **`_QSBThreadLocal()`**

```
template<typename _RAIter>
```

```
__gnu_parallel::__QSBThreadLocal< _RAIter >::_QSBThreadLocal (
    int __queue_size) [inline]
```

Constructor.

Parameters

|                           |                                  |
|---------------------------|----------------------------------|
| <code>__queue_size</code> | size of the work-stealing queue. |
|---------------------------|----------------------------------|

References [\\_M\\_leftover\\_parts](#).

### 5.142.4 Member Data Documentation

#### **`_M_elements_leftover`**

```
template<typename _RAIter>
```

```
volatile _DifferenceType* __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_elements_leftover
```

Pointer to a counter of elements left over to sort.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

#### **`_M_global`**

```
template<typename _RAIter>
```

```
_Piece __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_global
```

The complete sequence to sort.

#### **`_M_initial`**

```
template<typename _RAIter>
```

```
_Piece __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_initial
```

Initial piece to work on.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

#### **`_M_leftover_parts`**

```
template<typename _RAIter>
```

```
_RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_↵
leftover_parts
```

Work-stealing queue.

Referenced by [\\_QSBThreadLocal\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

**\_M\_num\_threads**

```
template<typename _RAIter>
_ThreadIndex __gnu_parallel::__qsbThreadLocal<_RAIter>::_M_num_threads
```

Number of threads involved in this algorithm.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

The documentation for this struct was generated from the following file:

- [balanced\\_quicksort.h](#)

**5.143 std::\_\_detail::\_Quoted\_string<\_String,\_CharT> Struct Template Reference**

```
#include <quoted_string.h>
```

**Public Member Functions**

- [\\_Quoted\\_string](#) (\_String \_\_str, \_CharT \_\_del, \_CharT \_\_esc)
- [\\_Quoted\\_string](#) & **operator=** ([\\_Quoted\\_string](#) &)=delete

**Public Attributes**

- \_CharT **\_M\_delim**
- \_CharT **\_M\_escape**
- \_String **\_M\_string**

**5.143.1 Detailed Description**

```
template<typename _String, typename _CharT>
struct std::__detail::_Quoted_string<_String,_CharT>
```

Struct for delimited strings.

The documentation for this struct was generated from the following file:

- [quoted\\_string.h](#)

**5.144 \_\_gnu\_parallel::\_RandomNumber Class Reference**

```
#include <random_number.h>
```

**Public Member Functions**

- [\\_RandomNumber](#) ()
- [\\_RandomNumber](#) (uint32\_t \_\_seed, uint64\_t \_M\_supremum=0x100000000ULL)
- unsigned long [\\_\\_genrand\\_bits](#) (int \_\_bits)
- uint32\_t [operator\(\)](#) ()
- uint32\_t [operator\(\)](#) (uint64\_t local\_supremum)

**5.144.1 Detailed Description**

Random number generator, based on the Mersenne twister.

**5.144.2 Constructor & Destructor Documentation**

```
\_RandomNumber() [1/2]
```

```
__gnu_parallel::_RandomNumber::_RandomNumber () [inline]
```

Default constructor. Seed with 0.

**\_RandomNumber()** [2/2]

```
__gnu_parallel::_RandomNumber::_RandomNumber (
    uint32_t __seed,
    uint64_t _M_supremum = 0x100000000ULL) [inline]
```

Constructor.

**Parameters**

|                          |                                                                                |
|--------------------------|--------------------------------------------------------------------------------|
| <code>__seed</code>      | Random <code>__seed</code> .                                                   |
| <code>_M_supremum</code> | Generate integer random numbers in the interval <code>[0,_M_supremum)</code> . |

**5.144.3 Member Function Documentation****\_\_genrand\_bits()**

```
unsigned long __gnu_parallel::_RandomNumber::_genrand_bits (
    int __bits) [inline]
```

Generate a number of random bits, run-time parameter.

**Parameters**

|                     |                             |
|---------------------|-----------------------------|
| <code>__bits</code> | Number of bits to generate. |
|---------------------|-----------------------------|

**operator>()** [1/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() () [inline]
```

Generate unsigned random 32-bit integer.

**operator>()** [2/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() (
    uint64_t local_supremum) [inline]
```

Generate unsigned random 32-bit integer in the interval `[0,local_supremum)`.

The documentation for this class was generated from the following file:

- [random\\_number.h](#)

**5.145 \_\_gnu\_cxx::\_Relative\_pointer\_impl<\_Tp> Class Template Reference**

```
#include <pointer.h>
```

Inheritance diagram for `__gnu_cxx::_Relative_pointer_impl<_Tp>`:



### Public Types

- typedef `_Tp` **element\_type**

### Public Member Functions

- `_Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__arg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__arg) const`
- `void set (_Tp *__arg)`

#### 5.145.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl<_Tp>
```

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

The documentation for this class was generated from the following file:

- [pointer.h](#)

### 5.146 `__gnu_cxx::_Relative_pointer_impl< const_Tp >` Class Template Reference

```
#include <pointer.h>
```

Inheritance diagram for `__gnu_cxx::_Relative_pointer_impl< const_Tp >`:



#### Public Types

- `typedef _Tp element_type`
- `typedef const_Tp element_type`

#### Public Member Functions

- `_Tp * get () const`
- `const _Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__arg) const`
- `bool operator< (const \_Relative\_pointer\_impl &__arg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__arg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__arg) const`
- `void set (_Tp *__arg)`
- `void set (const _Tp *__arg)`

#### 5.146.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl< const_Tp >
```

`Relative_pointer_impl` needs a specialization for `const T` because of the casting done during pointer arithmetic. The documentation for this class was generated from the following file:

- [pointer.h](#)

### 5.147 `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >` Class Template Reference

```
#include <queue.h>
```

## Public Member Functions

- [\\_RestrictedBoundedConcurrentQueue](#) ([\\_SequenceIndex](#) `__max_size`)
- [~\\_RestrictedBoundedConcurrentQueue](#) ()
- [pop\\_back](#) ([\\_Tp](#) & `__t`)
- [pop\\_front](#) ([\\_Tp](#) & `__t`)
- [push\\_front](#) (const [\\_Tp](#) & `__t`)

### 5.147.1 Detailed Description

`template<typename _Tp>`

`class __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>_Tp</code> | Contained element type. |
|------------------|-------------------------|

### 5.147.2 Constructor & Destructor Documentation

#### `_RestrictedBoundedConcurrentQueue()`

`template<typename _Tp>`

`__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::_RestrictedBoundedConcurrentQueue (`  
`\_SequenceIndex __max_size) [inline]`

Constructor. Not to be called concurrent, of course.

#### Parameters

|                         |                                             |
|-------------------------|---------------------------------------------|
| <code>__max_size</code> | Maximal number of elements to be contained. |
|-------------------------|---------------------------------------------|

References [\\_\\_gnu\\_parallel::\\_\\_encode2\(\)](#).

#### `~_RestrictedBoundedConcurrentQueue()`

`template<typename _Tp>`

`__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::~~_RestrictedBoundedConcurrentQueue ()`  
`[inline]`

Destructor. Not to be called concurrent, of course.

### 5.147.3 Member Function Documentation

#### `pop_back()`

`template<typename _Tp>`

`bool __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_back (`  
`\_Tp & __t) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with `pop_front()`.

References [\\_\\_gnu\\_parallel::\\_\\_compare\\_and\\_swap\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_decode2\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_encode2\(\)](#).

**pop\_front()**

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front (
    _Tp & __t) [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with pop\_front().

References [\\_\\_gnu\\_parallel::\\_\\_compare\\_and\\_swap\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_decode2\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_encode2\(\)](#).

**push\_front()**

```
template<typename _Tp>
void __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front (
    const _Tp & __t) [inline]
```

Pushes one element into the queue at the front end. Must not be called concurrently with pop\_front().

References [\\_\\_gnu\\_parallel::\\_\\_decode2\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_encode2\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_fetch\\_and\\_add\(\)](#).

The documentation for this class was generated from the following file:

- [queue.h](#)

**5.148 \_\_gnu\_debug::\_Safe\_container< \_SafeContainer, \_Alloc, \_SafeBase, \_IsCxx11AllocatorAware > Class Template Reference**

```
#include <safe_container.h>
```

Inheritance diagram for `__gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware`

>:



### Protected Member Functions

- `_Safe_container` (`_Safe_container` &&)=default
- `constexpr _Safe_container` (`_Safe_container` &&\_\_x, const `_Alloc` &\_\_a)
- `_Safe_container` (const `_Safe_container` &)=default



- constexpr void **\_M\_swap** (const [\\_Safe\\_container](#) &\_\_x) const noexcept
- constexpr [\\_Safe\\_container](#) & **operator=** ([\\_Safe\\_container](#) &&\_\_x) noexcept
- constexpr [\\_Safe\\_container](#) & **operator=** (const [\\_Safe\\_container](#) &) noexcept

#### 5.148.1 Detailed Description

```
template<typename _SafeContainer, typename _Alloc, template< typename > class _SafeBase, bool _IsCxx11AllocatorAware = true>
class __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >
```

Safe class dealing with some allocator dependent operations.

The documentation for this class was generated from the following file:

- [safe\\_container.h](#)

### 5.149 [\\_\\_gnu\\_debug::\\_Safe\\_forward\\_list](#)< [\\_SafeSequence](#) > Class Template Reference

```
#include <forward_list>
```

Inheritance diagram for [\\_\\_gnu\\_debug::\\_Safe\\_forward\\_list](#)< [\\_SafeSequence](#) >:



#### Public Member Functions

- void **\_M\_invalidate\_if** ([\\_Predicate](#) \_\_pred) const
- void **\_M\_transfer\_from\_if** (const [\\_Safe\\_sequence](#) &\_\_from, [\\_Predicate](#) \_\_pred) const

#### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

#### Protected Member Functions

- void **\_M\_detach\_all** () const
- void **\_M\_detach\_singular** () const
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & **\_M\_get\_mutex** () const noexcept
- void **\_M\_invalidate\_all** () const
- void **\_M\_revalidate\_singular** () const
- void **\_M\_swap** (const [\\_Safe\\_forward\\_list](#) &) const noexcept
- void **\_M\_swap** (const [\\_Safe\\_sequence\\_base](#) &\_\_x) const noexcept

#### 5.149.1 Detailed Description

```
template<typename _SafeSequence>
class __gnu_debug::_Safe_forward_list< _SafeSequence >
```

Special iterators swap and invalidation for `forward_list` because of the `before_begin` iterator.

### 5.149.2 Member Function Documentation

#### `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const [protected], [inherited]
```

Detach all iterators, leaving them singular.

#### `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### `_M_get_mutex()`

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]
```

For use in `_Safe_sequence`.

#### `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence<_SafeSequence>::_M_invalidate_if (
    _Predicate __pred) const [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

References [\\_\\_gnu\\_debug::\\_Safe\\_iterator\\_base::\\_M\\_next](#), and [\\_\\_gnu\\_debug::\\_Safe\\_iterator\\_base::\\_M\\_prior](#).

#### `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    const _Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence<_SafeSequence>::_M_transfer_from_if (
    const _Safe_sequence<_SafeSequence> & __from,
    _Predicate __pred) const [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

### 5.149.3 Member Data Documentation

#### `_M_const_iterators`

```
\_Safe\_iterator\_base\* __gnu_debug::_Safe_sequence_base::_M_const_iterators [mutable], [inherited]
```

The list of constant iterators that reference this container.

## **\_M\_iterators**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [mutable], [inherited]
```

The list of mutable iterators that reference this container.

## **\_M\_version**

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

## **5.150 \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, \_Category > Class Template Reference**

```
#include <safe_iterator.h>
```

Inheritance diagram for `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >`:



### **Public Types**

- typedef `_Safe_iterator`< `_Iterator`, `_Sequence`, `iterator_category` > **\_Self**
- typedef `_Traits::difference_type` **difference\_type**
- typedef `_Traits::iterator_category` **iterator\_category**
- typedef `_Iterator` **iterator\_type**
- typedef `_Traits::pointer` **pointer**
- typedef `_Traits::reference` **reference**
- typedef `_Traits::value_type` **value\_type**

### **Public Member Functions**

- constexpr `_Safe_iterator` () noexcept
- constexpr `_Safe_iterator` (`_Iterator` \_\_i, const `_Safe_sequence_base` \*\_\_seq) noexcept
- constexpr `_Safe_iterator` (`_Safe_iterator` &&\_\_x) noexcept
- constexpr `_Safe_iterator` (const `_Safe_iterator` &\_\_x) noexcept

- `template<typename _MutableIterator>`  
`constexpr __Safe_iterator (const __Safe_iterator<_MutableIterator, _Sequence, typename __gnu_cxx::__enable_if<_IsConstant::__value &&std::__are_same<_MutableIterator, _OtherIterator>::__value, _Category>::__type> &__x) noexcept`
- `void __M_attach (const __Safe_sequence_base * __seq)`
- `void __M_attach_single (const __Safe_sequence_base * __seq)`
- `bool __M_attached_to (const __Safe_sequence_base * __seq) const`
- `bool __M_before_dereferenceable () const`
- `template<typename _Diff>`  
`bool __M_can_advance (const std::pair<_Diff, _Distance_precision> &__dist, int __way) const`
- `bool __M_can_advance (difference_type __n, bool __strict=false) const`
- `bool __M_can_compare (const __Safe_iterator_base &__x) const noexcept`
- `bool __M_dereferenceable () const`
- `void __M_detach_single () noexcept`
- `_Distance_traits<_Iterator>::__type __M_get_distance_from_begin () const`
- `_Distance_traits<_Iterator>::__type __M_get_distance_to (const __Safe_iterator &__rhs) const`
- `_Distance_traits<_Iterator>::__type __M_get_distance_to_end () const`
- `__gnu_cxx::__conditional_type<_IsConstant::__value, const _Sequence *, _Sequence * >::__type __M_get_sequence () const`
- `bool __M_incrementable () const`
- `void __M_invalidate ()`
- `bool __M_is_before_begin () const`
- `constexpr bool __M_is_begin () const`
- `bool __M_is_beginnest () const`
- `bool __M_is_end () const`
- `void __M_reset () noexcept`
- `bool __M_singular () const noexcept`
- `void __M_unlink () noexcept`
- `bool __M_valid_range (const __Safe_iterator &__rhs, std::pair< difference_type, _Distance_precision> &__dist, bool __check_dereferenceable=true) const`
- `bool __M_value_initialized () const`
- `constexpr const _Iterator & base () const noexcept`
- `constexpr _Iterator & base () noexcept`
- `constexpr operator _Iterator () const noexcept`
- `constexpr reference operator* () const noexcept`
- `constexpr __Safe_iterator & operator++ () noexcept`
- `constexpr __Safe_iterator operator++ (int) noexcept`
- `constexpr pointer operator-> () const noexcept`
- `constexpr __Safe_iterator & operator= (__Safe_iterator &&__x) noexcept`
- `constexpr __Safe_iterator & operator= (const __Safe_iterator &__x) noexcept`

### Static Public Member Functions

- `static constexpr bool __S_constant ()`

### Public Attributes

- `__Safe_iterator_base * __M_next`
- `__Safe_iterator_base * __M_prior`
- `const __Safe_sequence_base * __M_sequence`
- `unsigned int __M_version`

## Protected Types

- typedef std::\_\_are\_same< typename \_Sequence::Base::const\_iterator, \_Iterator > **\_IsConstant**
- typedef \_\_gnu\_cxx::\_\_conditional\_type< \_IsConstant::\_\_value, typename \_Sequence::Base::iterator, typename \_Sequence::Base::const\_iterator >::\_\_type **\_OtherIterator**

## Protected Member Functions

- constexpr **\_Safe\_iterator** (const **\_Safe\_iterator** &\_\_x, \_Unchecked) noexcept
- void **\_M\_attach** (const **\_Safe\_sequence\_base** \* \_\_seq, bool \_\_constant)
- void **\_M\_attach\_single** (const **\_Safe\_sequence\_base** \* \_\_seq, bool \_\_constant) noexcept
- void **\_M\_detach** ()
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () noexcept

## Friends

- template<typename \_IterR>  
constexpr friend bool **operator==** (const **\_Self** &\_\_lhs, const **\_Safe\_iterator**< \_IterR, \_Sequence, iterator\_category > &\_\_rhs) noexcept
- constexpr friend bool **operator==** (const **\_Self** &\_\_lhs, const **\_Self** &\_\_rhs) noexcept

### 5.150.1 Detailed Description

template<typename \_Iterator, typename \_Sequence, typename \_Category = typename std::iterator\_traits<\_Iterator>::iterator\_category>

class \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, \_Category >

Safe iterator wrapper.

The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Note that `_Iterator` must be the first base class so that it gets initialized before the iterator is being attached to the container's list of iterators and it is being detached before `_Iterator` get destroyed. Otherwise it would result in a data race.

### 5.150.2 Constructor & Destructor Documentation

**\_Safe\_iterator()** [1/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator () [inline],
[constexpr], [noexcept]
```

**Postcondition**

the iterator is singular and unattached

**\_Safe\_iterator()** [2/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
    _Iterator __i,
    const _Safe_sequence_base * __seq) [inline], [constexpr], [noexcept]
```

Safe iterator construction from an unsafe iterator and its sequence.

**Precondition**

`seq` is not NULL

**Postcondition**

this is not singular

**`_Safe_iterator()` [3/5]**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_Safe_iterator (
    const __Safe_iterator<_Iterator, _Sequence, _Category> & __x) [inline], [constexpr],
[noexcept]
Copy construction.
```

**`_Safe_iterator()` [4/5]**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_Safe_iterator (
    __Safe_iterator<_Iterator, _Sequence, _Category> && __x) [inline], [constexpr],
[noexcept]
Move construction.
```

**Postcondition**

`__x` is singular and unattached

**`_Safe_iterator()` [5/5]**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
template<typename _MutableIterator>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_Safe_iterator (
    const __Safe_iterator<_MutableIterator, _Sequence, typename __gnu_cxx::__enable_if<
_IsConstant::__value &&std::__are_same<_MutableIterator, _OtherIterator>::__value, _Category>::__type> & __x) [inline], [constexpr], [noexcept]
Converting constructor from a mutable iterator to a constant iterator.
```

**5.150.3 Member Function Documentation****`_M_attach()` [1/2]**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
void __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_M_attach (
    const __Safe_sequence_base * __seq) [inline]
Attach iterator to the given sequence.
```

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()` and `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`.

**\_M\_attach()** [2/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
    const _Safe_sequence_base * __seq,
    bool __constant) [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References [\\_M\\_attach\(\)](#).

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_M\\_attach\(\)](#), and [\\_M\\_attach\(\)](#).

**\_M\_attach\_single()** [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
void __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >::_M_attach_single (
    const _Safe_sequence_base * __seq) [inline]
```

Likewise, but not thread-safe.

**\_M\_attach\_single()** [2/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
    const _Safe_sequence_base * __seq,
    bool __constant) [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

References [\\_M\\_attach\\_single\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_M\\_attach\\_single\(\)](#), and [\\_M\\_attach\\_single\(\)](#).

**\_M\_attached\_to()**

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
    const _Safe_sequence_base * __seq) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

References [\\_M\\_attached\\_to\(\)](#), and [\\_M\\_sequence](#).

Referenced by [\\_M\\_attached\\_to\(\)](#).

**\_M\_before\_dereferenceable()**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >::_M_before_dereferenceable ()
const [inline]
```

Is the iterator before a dereferenceable one?

**\_M\_can\_compare()**

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
    const _Safe_iterator_base & __x) const [noexcept], [inherited]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

References [\\_Safe\\_iterator\\_base\(\)](#), and [\\_M\\_can\\_compare\(\)](#).

Referenced by [\\_M\\_can\\_compare\(\)](#).

**`_M_dereferenceable()`**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator*()`, and `__gnu_debug::_Safe_iterator< _Base_iterator, m`

**`_M_detach()`**

```
void __gnu_debug::_Safe_iterator_base::_M_detach () [protected], [inherited]
```

Detach the iterator for whatever sequence it is attached to, if any.

References `_M_detach()`.

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator()`, `_M_detach()`, `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=()`, and `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=()`.

**`_M_detach_single()`**

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single () [noexcept], [inherited]
```

Likewise, but not thread-safe.

**`_M_get_mutex()`**

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex () [protected], [noexcept],
[inherited]
```

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++()`, `__gnu_debug::_Safe_local_iterator< _OtherIterator, map >::operator++()`, `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=()`, `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=()`, `__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=()`, and `__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=()`.

**`_M_incrementable()`**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_incrementable () const
[inline]
```

Is the iterator incrementable?

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_before_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++()`, and `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++()`.

**`_M_invalidate()`**

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline], [inherited]
```

Invalidate the iterator, making it singular.

References `_M_invalidate()`, and `_M_version`.

Referenced by `_M_invalidate()`.

**`_M_is_before_begin()`**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_before_begin () const
[inline]
```

Is this iterator equal to the sequence's `before_begin()` iterator if any?

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable()`.





Is the iterator value-initialized?

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, and `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`.

### `_S_constant()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
static constexpr bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_S_constant() [inline], [static], [constexpr]
```

Determine if this is a constant iterator.

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_attach()`, and `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_attach()`.

### `base()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Iterator & __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::base() [inline], [constexpr], [noexcept]
```

Return the underlying iterator.

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_is_begin()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_is_end()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_value_initialized()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator++()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator--()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, and `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`.

### `operator_iterator()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::operator_iterator() const [inline], [constexpr], [noexcept]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

### `operator*()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
reference __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::operator*() const [inline], [nodiscard], [constexpr], [noexcept]
```

Iterator dereference.

#### Precondition

iterator is dereferenceable

### `operator++()` [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::operator++() [inline], [constexpr], [noexcept]
```

Iterator preincrement.

**Precondition**

iterator is incrementable

**operator++()** [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator++ (
    int ) [inline], [constexpr], [noexcept]
```

Iterator postincrement.

**Precondition**

iterator is incrementable

**operator->()**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
pointer __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator-> () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Iterator dereference.

**Precondition**

iterator is dereferenceable

**operator=()** [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
    _Safe_iterator< _Iterator, _Sequence, _Category > && __x) [inline], [constexpr],
[noexcept]
```

Move assignment.

**Postcondition**

\_\_x is singular and unattached

**operator=()** [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
    const _Safe_iterator< _Iterator, _Sequence, _Category > & __x) [inline], [constexpr],
[noexcept]
```

Copy assignment.

**5.150.4 Member Data Documentation****\_M\_next**

[\\_Safe\\_iterator\\_base\\*](#) [\\_\\_gnu\\_debug::\\_Safe\\_iterator\\_base::\\_M\\_next](#) [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence< \\_SafeSequence >](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence< \\_Sequence >::\\_M\\_transfer\\_from\\_if\(\)](#), and [\\_M\\_unlink\(\)](#).

**`_M_prior`**

`_Safe_iterator_base* __gnu_debug::Safe_iterator_base::_M_prior` [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_Safe_iterator_base()`, `__gnu_debug::Safe_sequence<_SafeSequence>::_M_transfer_from_if()`, and `_M_unlink()`.

**`_M_sequence`**

`const _Safe_sequence_base* __gnu_debug::Safe_iterator_base::_M_sequence` [inherited]

The sequence this iterator references; may be `NULL` to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by `__gnu_debug::Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_M_attached_to()`, `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`, `__gnu_debug::Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`, and `__gnu_debug::Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`.

**`_M_version`**

`unsigned int __gnu_debug::Safe_iterator_base::_M_version` [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Referenced by `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_M_invalidate()`, and `__gnu_debug::Safe_sequence<_SafeSequence>::_M_transfer_from_if()`.

The documentation for this class was generated from the following files:

- `boost_concept_check.h`
- `safe_iterator.h`
- `safe_iterator.tcc`

**5.151 `__gnu_debug::Safe_iterator_base` Class Reference**

```
#include <safe_base.h>
```

## Public Member Functions

- `bool _M_attached_to (const _Safe_sequence_base * __seq) const`
- `bool _M_can_compare (const _Safe_iterator_base & __x) const noexcept`

- void `_M_detach_single` () noexcept
- void `_M_invalidate` ()
- void `_M_reset` () noexcept
- bool `_M_singular` () const noexcept
- void `_M_unlink` () noexcept

### Public Attributes

- `_Safe_iterator_base` \* `_M_next`
- `_Safe_iterator_base` \* `_M_prior`
- const `_Safe_sequence_base` \* `_M_sequence`
- unsigned int `_M_version`

### Protected Member Functions

- constexpr `_Safe_iterator_base` ()
- constexpr `_Safe_iterator_base` (const `_Safe_iterator_base` &\_\_x, bool \_\_constant)
- constexpr `_Safe_iterator_base` (const `_Safe_sequence_base` \* \_\_seq, bool \_\_constant)
- void `_M_attach` (const `_Safe_sequence_base` \* \_\_seq, bool \_\_constant)
- void `_M_attach_single` (const `_Safe_sequence_base` \* \_\_seq, bool \_\_constant) noexcept
- void `_M_detach` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () noexcept

### Friends

- class `_Safe_sequence_base`

#### 5.151.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

#### 5.151.2 Constructor & Destructor Documentation

##### `_Safe_iterator_base`() [1/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base () [inline], [constexpr], [protected]
```

Initializes the iterator and makes it singular.

References `_M_next`, `_M_prior`, `_M_sequence`, and `_M_version`.

Referenced by `_Safe_iterator_base()`, and `_M_can_compare()`.

##### `_Safe_iterator_base`() [2/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
    const _Safe_sequence_base * __seq,
    bool __constant) [inline], [constexpr], [protected]
```

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

References `_M_attach()`, `_M_next`, `_M_prior`, `_M_sequence`, and `_M_version`.

**`_Safe_iterator_base()`** [3/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
    const \_Safe\_iterator\_base & __x,
    bool __constant) [inline], [constexpr], [protected]
```

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

References [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_attach\(\)](#), [\\_M\\_next](#), [\\_M\\_prior](#), [\\_M\\_sequence](#), and [\\_M\\_version](#).

**5.151.3 Member Function Documentation****`_M_attach()`**

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
    const \_Safe\_sequence\_base * __seq,
    bool __constant) [protected]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References [\\_M\\_attach\(\)](#).

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_M\\_attach\(\)](#), and [\\_M\\_attach\(\)](#).

**`_M_attach_single()`**

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
    const \_Safe\_sequence\_base * __seq,
    bool __constant) [protected], [noexcept]
```

Likewise, but not thread-safe.

References [\\_M\\_attach\\_single\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_M\\_attach\\_single\(\)](#), and [\\_M\\_attach\\_single\(\)](#).

**`_M_attached_to()`**

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
    const \_Safe\_sequence\_base * __seq) const [inline]
```

Determines if we are attached to the given sequence.

References [\\_M\\_attached\\_to\(\)](#), and [\\_M\\_sequence](#).

Referenced by [\\_M\\_attached\\_to\(\)](#).

**`_M_can_compare()`**

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
    const \_Safe\_iterator\_base & __x) const [noexcept]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

References [\\_Safe\\_iterator\\_base\(\)](#), and [\\_M\\_can\\_compare\(\)](#).

Referenced by [\\_M\\_can\\_compare\(\)](#).

**`_M_detach()`**

```
void __gnu_debug::_Safe_iterator_base::_M_detach () [protected]
```

Detach the iterator for whatever sequence it is attached to, if any.

References [\\_M\\_detach\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_M\\_detach\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#).

## **\_M\_detach\_single()**

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single () [noexcept]
```

Likewise, but not thread-safe.

## **\_M\_get\_mutex()**

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex () [protected], [noexcept]
```

For use in `_Safe_iterator`.

Referenced by [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator<\\_Base\\_iterator, map >::operator++\(\)](#), [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_\\_gnu\\_debug::\\_\\_Safe\\_iterator<\\_Base\\_iterator, map >::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator<\\_Base\\_iterator, map >::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer >::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator<\\_OtherIterator](#)

## **`_M_invalidate()`**

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline]
```

Invalidate the iterator, making it singular.

References [\\_M\\_invalidate\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\(\)](#).

## **\_M\_reset()**

```
void __gnu_debug::_Safe_iterator_base::_M_reset () [noexcept]
```

### Reset all member variables

## **\_M\_singular()**

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const [noexcept]
```

Is this iterator singular?

```

Referenced by __gnu_debug:: Safe_iterator< _Base_iterator, map >:: Safe_iterator(), __gnu_debug:: Safe_iterator< _Base_iterator, m
__gnu_debug:: Safe_iterator< _Base_iterator, map >:: Safe_iterator(), __gnu_debug:: Safe_local_iterator< _OtherIterator, _UContain
__gnu_debug:: Safe_local_iterator< _OtherIterator, _UContainer >:: Safe_local_iterator(), __gnu_debug:: Safe_local_iterator< _Other
__gnu_debug:: check_singular_aux(), __gnu_debug:: Safe_iterator< _Base_iterator, map >:: M_dereferenceable(),
__gnu_debug:: Safe_local_iterator< _OtherIterator, _UContainer >:: M_dereferenceable(), __gnu_debug:: Safe_iterator< _Base_ite
__gnu_debug:: Safe_local_iterator< _OtherIterator, _UContainer >:: M_incrementable(), __gnu_debug:: Safe_iterator< _Base_iterator
__gnu_debug:: Safe_iterator< _Base_iterator, map >:: operator=(), __gnu_debug:: Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug:: Safe_local_iterator< _OtherIterator, _UContainer >:: operator=().

```

## M unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () [inline], [noexcept]
```

Unlink itself

References `_M_next`, `_M_prior`, and `_M_unlink()`.

Referenced by [M](#) [unlink\(\)](#).

#### 5.151.4 Member Data Documentation

## \_M\_next

```
__Safe_iterator_base* __gnu_debug::__Safe_iterator_base::_M_next
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_SafeSequence>](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<Sequence>::M transfer from if\(\)](#), and [M unlink\(\)](#).



**\_M\_prior**

```
\_Safe\_iterator\_base\* __gnu_debug::_Safe_iterator_base::_M_prior
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_SafeSequence>::\\_M\\_transfer\\_from\\_if\(\)](#), and [\\_M\\_unlink\(\)](#).

**\_M\_sequence**

```
const \_Safe\_sequence\_base\* __gnu_debug::_Safe_iterator_base::_M_sequence
```

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_attached\\_to\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#).

**\_M\_version**

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_invalidate\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_SafeSequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

## 5.152 `__gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>` Class Template Reference

```
#include <safe_local_iterator.h>
```

Inheritance diagram for `__gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>`:



### Public Types

- `typedef _Traits::difference_type` **difference\_type**
- `typedef _Traits::iterator_category` **iterator\_category**
- `typedef _Iterator` **iterator\_type**
- `typedef _Traits::pointer` **pointer**
- `typedef _Traits::reference` **reference**
- `typedef _Traits::value_type` **value\_type**

### Public Member Functions

- `_Safe_local_iterator` () noexcept
- `_Safe_local_iterator` (\_Iterator \_\_i, const `_Safe_unordered_container_base` \*\_\_cont)
- `_Safe_local_iterator` (`_Safe_local_iterator` &&\_\_x) noexcept
- `_Safe_local_iterator` (const `_Safe_local_iterator` &\_\_x) noexcept
- `template<typename _MutableIterator>`  
`_Safe_local_iterator` (const `_Safe_local_iterator`<\_MutableIterator, typename `__gnu_cxx::__enable_if`<\_IsConstant::\_\_value &&std::is\_same<\_MutableIterator, \_OtherIterator>::\_\_value, \_UContainer>::\_\_type> &\_\_x) noexcept
- `void` `_M_attach` (const `_Safe_unordered_container_base` \*\_\_cont)
- `void` `_M_attach_single` (const `_Safe_unordered_container_base` \*\_\_cont)
- `bool` `_M_attached_to` (const `_Safe_sequence_base` \*\_\_seq) const
- `bool` `_M_can_compare` (const `_Safe_iterator_base` &\_\_x) const noexcept
- `bool` `_M_dereferenceable` () const
- `_Distance_traits`<\_Iterator>::\_\_type `_M_get_distance_to` (const `_Safe_local_iterator` &\_\_rhs) const
- `std::conditional_t`<\_IsConstant::\_\_value, const \_UContainer \*, \_UContainer \* > `_M_get_ucontainer` () const

- `template<typename _Other>`  
`bool _M_in_same_bucket (const _Safe_local_iterator< _Other, _UContainer > &__other) const`
- `bool _M_incrementable () const`
- `void _M_invalidate ()`
- `bool _M_is_begin () const`
- `bool _M_is_end () const`
- `void _M_reset () noexcept`
- `const _Safe_unordered_container_base * _M_safe_container () const noexcept`
- `bool _M_singular () const noexcept`
- `void _M_unlink () noexcept`
- `bool _M_valid_range (const _Safe_local_iterator &__rhs, std::pair< difference_type, _Distance_precision > &__dist_info) const`
- `bool _M_value_initialized () const`
- `const _Iterator & base () const noexcept`
- `_Iterator & base () noexcept`
- `size_type bucket () const`
- `operator _Iterator () const`
- `reference operator* () const`
- `_Safe_local_iterator & operator++ ()`
- `_Safe_local_iterator operator++ (int)`
- `pointer operator-> () const`
- `_Safe_local_iterator & operator= (_Safe_local_iterator &&__x) noexcept`
- `_Safe_local_iterator & operator= (const _Safe_local_iterator &__x)`

### Static Public Member Functions

- `static constexpr bool _S_constant ()`

### Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `const _Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_attach (const _Safe_sequence_base *__seq, bool __constant)`
- `void _M_attach (const _Safe_unordered_container_base *__cont, bool __constant)`
- `void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept`
- `void _M_attach_single (const _Safe_unordered_container_base *__cont, bool __constant) noexcept`
- `void _M_detach ()`
- `void _M_detach_single () noexcept`
- `__gnu_cxx::__mutex & _M_get_mutex () noexcept`

### Friends

- `bool operator!= (const _Self &__lhs, const _OtherSelf &__rhs) noexcept`
- `bool operator!= (const _Self &__lhs, const _Self &__rhs) noexcept`
- `bool operator== (const _Self &__lhs, const _OtherSelf &__rhs) noexcept`
- `bool operator== (const _Self &__lhs, const _Self &__rhs) noexcept`

### 5.152.1 Detailed Description

`template<typename _Iterator, typename _UContainer>`  
**class** `__gnu_debug::__Safe_local_iterator<_Iterator, _UContainer>`

Safe iterator wrapper.

The class template `_Safe_local_iterator` is a wrapper around an iterator that tracks the iterator's movement among unordered containers and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_local_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from unordered containers, and querying the iterator's state.

### 5.152.2 Constructor & Destructor Documentation

#### `_Safe_local_iterator()` [1/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::__Safe_local_iterator<_Iterator, _UContainer>::__Safe_local_iterator () [inline],
[noexcept]
```

##### Postcondition

the iterator is singular and unattached

#### `_Safe_local_iterator()` [2/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::__Safe_local_iterator<_Iterator, _UContainer>::__Safe_local_iterator (
    _Iterator __i,
    const __Safe_unordered_container_base * __cont) [inline]
```

Safe iterator construction from an unsafe iterator and its unordered container.

##### Precondition

`cont` is not NULL

##### Postcondition

this is not singular

#### `_Safe_local_iterator()` [3/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::__Safe_local_iterator<_Iterator, _UContainer>::__Safe_local_iterator (
    const __Safe_local_iterator<_Iterator, _UContainer> & __x) [inline], [noexcept]
```

Copy construction.

#### `_Safe_local_iterator()` [4/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::__Safe_local_iterator<_Iterator, _UContainer>::__Safe_local_iterator (
    __Safe_local_iterator<_Iterator, _UContainer> && __x) [inline], [noexcept]
```

Move construction.

##### Postcondition

`__x` is singular and unattached

**`_Safe_local_iterator()`** [5/5]

```
template<typename _Iterator, typename _UContainer>
template<typename _MutableIterator>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
    const __Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< _↵
IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _UContainer
>::__type > & __x) [inline], [noexcept]
```

Converting constructor from a mutable iterator to a constant iterator.

**5.152.3 Member Function Documentation****`_M_attach()`** [1/3]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
    const __Safe_sequence_base * __seq,
    bool __constant) [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References [\\_M\\_attach\(\)](#).

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_M\\_attach\(\)](#), and [\\_M\\_attach\(\)](#).

**`_M_attach()`** [2/3]

```
template<typename _Iterator, typename _UContainer>
void __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_attach (
    const __Safe_unordered_container_base * __cont) [inline]
```

Attach iterator to the given unordered container.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#).

**`_M_attach()`** [3/3]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
    const __Safe_unordered_container_base * __cont,
    bool __constant) [protected], [inherited]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by [\\_Safe\\_local\\_iterator\\_base\(\)](#), [\\_Safe\\_local\\_iterator\\_base\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#).

**`_M_attach_single()`** [1/3]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
    const __Safe_sequence_base * __seq,
    bool __constant) [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

References [\\_M\\_attach\\_single\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_M\\_attach\\_single\(\)](#), and [\\_M\\_attach\\_single\(\)](#).

**`_M_attach_single()`** [2/3]

```
template<typename _Iterator, typename _UContainer>
void __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_attach_single (
    const __Safe_unordered_container_base * __cont) [inline]
```

Likewise, but not thread-safe.

### `_M_attach_single()` [3/3]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
    const __Safe_unordered_container_base * __cont,
    bool __constant) [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::_M_attach_single()`.

### `_M_attached_to()`

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
    const __Safe_sequence_base * __seq) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

References `_M_attached_to()`, and `_M_sequence`.

Referenced by `_M_attached_to()`.

### `_M_can_compare()`

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
    const __Safe_iterator_base & __x) const [noexcept], [inherited]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

References `_Safe_iterator_base()`, and `_M_can_compare()`.

Referenced by `_M_can_compare()`.

### `_M_dereferenceable()`

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?

Referenced by `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator*()`, and `__gnu_debug::_Safe_local_iterator`.

### `_M_detach()`

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach () [protected], [inherited]
```

Detach the iterator for whatever container it is attached to, if any.

Referenced by `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::_Safe_local_iterator()`, `__gnu_debug::_Safe_local_iterator`, and `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`.

### `_M_detach_single()`

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single () [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

### `_M_get_mutex()`

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex () [protected], [noexcept],
[inherited]
```

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator++()`, `__gnu_debug::_Safe_local_iterator<_OtherIterator, __gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`, and `__gnu_debug::_Safe_local_iterator<_OtherIterator`.

### **\_M\_in\_same\_bucket()**

```
template<typename _Iterator, typename _UContainer>
template<typename _Other>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_in_same_bucket (
    const __Safe_local_iterator< _Other, _UContainer > & __other) const [inline]
```

Is this iterator part of the same bucket as the other one?

### **\_M\_incrementable()**

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_incrementable () const [inline]
```

Is the iterator incrementable?

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator++\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)

### **\_M\_invalidate()**

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline], [inherited]
```

Invalidate the iterator, making it singular.

References [\\_M\\_invalidate\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\(\)](#).

### **\_M\_is\_begin()**

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_is_begin () const [inline]
```

Is this iterator equal to the container's begin(bucket) iterator?

### **\_M\_is\_end()**

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_is_end () const [inline]
```

Is this iterator equal to the container's end(bucket) iterator?

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_M\\_dereferenceable\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_M\\_incrementable\(\)](#).

### **\_M\_reset()**

```
void __gnu_debug::_Safe_iterator_base::_M_reset () [noexcept], [inherited]
```

Reset all member variables

### **\_M\_singular()**

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const [noexcept], [inherited]
```

Is this iterator singular?

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_check\\_singular\\_aux\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_M\\_dereferenceable\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_M\\_dereferenceable\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_M\\_incrementable\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#).

**`_M_unlink()`**

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () [inline], [noexcept], [inherited]
```

Unlink itself

References [\\_M\\_next](#), [\\_M\\_prior](#), and [\\_M\\_unlink\(\)](#).

Referenced by [\\_M\\_unlink\(\)](#).

**`_M_value_initialized()`**

```
template<typename _Iterator, typename _UContainer>
```

```
bool __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::_M_value_initialized () const  
[inline]
```

Is the iterator value-initialized?

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#).

**`_S_constant()`**

```
template<typename _Iterator, typename _UContainer>
```

```
static constexpr bool __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::_S_constant ()  
[inline], [static], [constexpr]
```

Determine if this is a constant iterator.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_attach\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#).

**`base()`**

```
template<typename _Iterator, typename _UContainer>
```

```
_Iterator & __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::base () [inline], [noexcept]
```

Return the underlying iterator.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_is\\_end\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::bucket\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator++\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#).

**`bucket()`**

```
template<typename _Iterator, typename _UContainer>
```

```
size_type __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::bucket () const [inline]
```

Return the bucket.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_in\\_same\\_bucket\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_is\\_end\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_is\\_end\(\)](#).

**`operator_iterator()`**

```
template<typename _Iterator, typename _UContainer>
```

```
__gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::operator_iterator () const [inline]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

**`operator*()`**

```
template<typename _Iterator, typename _UContainer>
```

```
reference __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::operator* () const [inline]
```

Iterator dereference.



**Precondition**

iterator is dereferenceable

**operator++()** [1/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator++ ()
[inline]
```

Iterator preincrement.

**Precondition**

iterator is incrementable

**operator++()** [2/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator++ (
    int ) [inline]
```

Iterator postincrement.

**Precondition**

iterator is incrementable

**operator->()**

```
template<typename _Iterator, typename _UContainer>
pointer __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator-> () const [inline]
```

Iterator dereference.

**Precondition**

iterator is dereferenceable

**operator=()** [1/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator= (
    _Safe_local_iterator< _Iterator, _UContainer > && __x) [inline], [noexcept]
```

Move assignment.

**Postcondition**

\_\_x is singular and unattached

**operator=()** [2/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator= (
    const _Safe_local_iterator< _Iterator, _UContainer > & __x) [inline]
```

Copy assignment.

## 5.152.4 Member Data Documentation

**`_M_next`**

`_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next` [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_SafeSequence>](#)

[\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), and [\\_M\\_unlink\(\)](#).

**`_M_prior`**

`_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior` [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_SafeSequence>](#)

[\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), and [\\_M\\_unlink\(\)](#).

**`_M_sequence`**

`const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence` [inherited]

The sequence this iterator references; may be `NULL` to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as `const`.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, m](#)

[\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#),

[\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_attached\\_to\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#),

[\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#),

[\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator](#)

**`_M_version`**

`unsigned int __gnu_debug::_Safe_iterator_base::_M_version` [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_invalidate\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence](#)

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_local\\_iterator.h](#)
- [safe\\_local\\_iterator.tcc](#)

5.153 `__gnu_debug::_Safe_local_iterator_base` Class Reference

```
#include <safe_unordered_base.h>
```

Inheritance diagram for `__gnu_debug::_Safe_local_iterator_base`:



### Public Member Functions

- `bool _M_attached_to (const \_Safe\_sequence\_base * __seq) const`
- `bool _M_can_compare (const \_Safe\_iterator\_base & __x) const noexcept`
- `void _M_invalidate ()`

- void `_M_reset` () noexcept
- const `_Safe_unordered_container_base` \* `_M_safe_container` () const noexcept
- bool `_M_singular` () const noexcept
- void `_M_unlink` () noexcept

### Public Attributes

- `_Safe_iterator_base` \* `_M_next`
- `_Safe_iterator_base` \* `_M_prior`
- const `_Safe_sequence_base` \* `_M_sequence`
- unsigned int `_M_version`

### Protected Member Functions

- `_Safe_local_iterator_base` ()
- `_Safe_local_iterator_base` (const `_Safe_local_iterator_base` &\_\_x, bool \_\_constant)
- `_Safe_local_iterator_base` (const `_Safe_unordered_container_base` \*\_\_seq, bool \_\_constant)
- void `_M_attach` (const `_Safe_sequence_base` \*\_\_seq, bool \_\_constant)
- void `_M_attach` (const `_Safe_unordered_container_base` \*\_\_cont, bool \_\_constant)
- void `_M_attach_single` (const `_Safe_sequence_base` \*\_\_seq, bool \_\_constant) noexcept
- void `_M_attach_single` (const `_Safe_unordered_container_base` \*\_\_cont, bool \_\_constant) noexcept
- void `_M_detach` ()
- void `_M_detach_single` () noexcept
- `__gnu_cxx::__mutex` & `_M_get_mutex` () noexcept

#### 5.153.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_local_iterator_base` base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

#### 5.153.2 Constructor & Destructor Documentation

##### `_Safe_local_iterator_base`() [1/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base () [inline], [protected]
```

Initializes the iterator and makes it singular.

Referenced by `_Safe_local_iterator_base`().

##### `_Safe_local_iterator_base`() [2/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
    const _Safe_unordered_container_base * __seq,
    bool __constant) [inline], [protected]
```

Initialize the iterator to reference the container pointed to by `__seq`. `__constant` is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

References `_M_attach`().

**`_Safe_local_iterator_base()` [3/3]**

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
    const \_Safe\_local\_iterator\_base & __x,
    bool __constant) [inline], [protected]
```

Initializes the iterator to reference the same container that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

References [\\_Safe\\_local\\_iterator\\_base\(\)](#), and [\\_M\\_attach\(\)](#).

**5.153.3 Member Function Documentation****`_M_attach()` [1/2]**

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
    const \_Safe\_sequence\_base * __seq,
    bool __constant) [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References [\\_M\\_attach\(\)](#).

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_M\\_attach\(\)](#), and [\\_M\\_attach\(\)](#).

**`_M_attach()` [2/2]**

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
    const \_Safe\_unordered\_container\_base * __cont,
    bool __constant) [protected]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by [\\_Safe\\_local\\_iterator\\_base\(\)](#), [\\_Safe\\_local\\_iterator\\_base\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, UContainer >::\\_M\\_attach\(\)](#).

**`_M_attach_single()` [1/2]**

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
    const \_Safe\_sequence\_base * __seq,
    bool __constant) [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

References [\\_M\\_attach\\_single\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_M\\_attach\\_single\(\)](#), and [\\_M\\_attach\\_single\(\)](#).

**`_M_attach_single()` [2/2]**

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
    const \_Safe\_unordered\_container\_base * __cont,
    bool __constant) [protected], [noexcept]
```

Likewise, but not thread-safe.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, UContainer >::\\_M\\_attach\\_single\(\)](#).

**`_M_attached_to()`**

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
    const \_Safe\_sequence\_base * __seq) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

References [\\_M\\_attached\\_to\(\)](#), and [\\_M\\_sequence](#).

Referenced by [\\_M\\_attached\\_to\(\)](#).



## **`_M_unlink()`**

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () [inline], [noexcept], [inherited]
```

Unlink itself

References [\\_M\\_next](#), [\\_M\\_prior](#), and [\\_M\\_unlink\(\)](#).

Referenced by [\\_M\\_unlink\(\)](#).

### **5.153.4 Member Data Documentation**

## **`_M_next`**

```
\_Safe\_iterator\_base\* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_SafeSequence>::\\_M\\_next\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), and [\\_M\\_unlink\(\)](#).

## **`_M_prior`**

```
\_Safe\_iterator\_base\* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_SafeSequence>::\\_M\\_prior\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), and [\\_M\\_unlink\(\)](#).

## **`_M_sequence`**

```
const \_Safe\_sequence\_base\* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
```

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_attached\\_to\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#).

## **`_M_version`**

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_invalidate\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\(\)](#).

The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)

### **5.154 `__gnu_debug::_Safe_node_sequence<_Sequence>` Class Template Reference**

```
#include <safe_sequence.h>
```

Inheritance diagram for `__gnu_debug::_Safe_node_sequence<_Sequence>`:



### Public Member Functions

- `template<typename _Predicate>`  
`void _M_invalidate_if ( _Predicate __pred) const`
- `template<typename _Predicate>`  
`void _M_transfer_from_if (const _Safe_sequence &__from, _Predicate __pred) const`

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all () const`
- `void _M_detach_singular () const`
- `__gnu_cxx::__mutex & _M_get_mutex () const noexcept`
- `void _M_invalidate_all () const`
- `void _M_revalidate_singular () const`
- `void _M_swap (const _Safe_sequence_base &__x) const noexcept`

#### 5.154.1 Detailed Description

`template<typename _Sequence>`  
**class** `__gnu_debug::_Safe_node_sequence<_Sequence>`

Like `_Safe_sequence` but with a special `_M_invalidate_all` implementation not invalidating past-the-end iterators. Used by node based sequence.



### 5.154.2 Member Function Documentation

#### **`_M_detach_all()`**

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by [~\\_Safe\\_sequence\\_base\(\)](#).

#### **`_M_detach_singular()`**

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
```

Detach all singular iterators.

#### **Postcondition**

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

#### **`_M_get_mutex()`**

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]
```

For use in `_Safe_sequence`.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_local\\_if\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#).

#### **`_M_invalidate_if()`**

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if (
    _Predicate __pred) const [inherited]
```

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

References [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_const\\_iterators](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_get\\_mutex\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_iterators](#).

#### **`_M_revalidate_singular()`**

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### **`_M_swap()`**

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    const _Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### **`_M_transfer_from_if()`**

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if (
    const _Safe_sequence<_Sequence> & __from,
    _Predicate __pred) const [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

References `std::__addressof()`, `__gnu_debug::Safe_sequence_base::M_const_iterators`, `__gnu_debug::Safe_sequence_base::M_g`, `__gnu_debug::Safe_sequence_base::M_iterators`, `__gnu_debug::Safe_iterator_base::M_next`, `__gnu_debug::Safe_iterator_base::`, `__gnu_debug::Safe_iterator_base::M_sequence`, `__gnu_debug::Safe_iterator_base::M_version`, and `__gnu_debug::Safe_sequence`.

### 5.154.3 Member Data Documentation

#### `_M_const_iterators`

`_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` [mutable], [inherited]

The list of constant iterators that reference this container.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_invalidate_if()`, `__gnu_debug::Safe_unordered_container<_Conta` and `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

#### `_M_iterators`

`_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` [mutable], [inherited]

The list of mutable iterators that reference this container.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::M_invalidate_if()`, `__gnu_debug::Safe_unordered_container<_Conta` and `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`.

#### `_M_version`

`unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Referenced by `_M_invalidate_all()`, `__gnu_debug::Safe_sequence<_Sequence>::M_transfer_from_if()`, `__gnu_debug::Safe_iterator`, `__gnu_debug::Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::Safe_local_iterator<_OtherIterator, _UContainer>` and `__gnu_debug::Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`.

The documentation for this class was generated from the following file:

- `safe_sequence.h`

### 5.155 `__gnu_debug::Safe_sequence<_Sequence>` Class Template Reference

```
#include <safe_sequence.h>
```

Inheritance diagram for `__gnu_debug::_Safe_sequence<_Sequence>`:



## Public Member Functions

- `template<typename _Predicate>`  
`void _M_invalidate_if (_Predicate __pred) const`
- `template<typename _Predicate>`  
`void _M_transfer_from_if (const _Safe_sequence &__from, _Predicate __pred) const`

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `void _M_detach_all () const`
- `void _M_detach_singular () const`
- `__gnu_cxx::__mutex & _M_get_mutex () const noexcept`
- `void _M_invalidate_all () const`
- `void _M_revalidate_singular () const`
- `void _M_swap (const _Safe_sequence_base &__x) const noexcept`

### 5.155.1 Detailed Description

```

template<typename _Sequence>
class __gnu_debug::_Safe_sequence<_Sequence>

```

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template `_Safe_sequence` simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this sequence. Iterators will then be tracked automatically.

### 5.155.2 Member Function Documentation

#### `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by [~\\_Safe\\_sequence\\_base\(\)](#).

#### `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### `_M_get_mutex()`

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]
```

For use in `_Safe_sequence`.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_local\\_if\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_all\(\)](#).

#### `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

References [\\_M\\_invalidate\\_if\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\\_all\(\)](#).

#### `_M_invalidate_if()`

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if (
    _Predicate __pred) const
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

References [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_const\\_iterators](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_get\\_mutex\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_iterators](#).

#### `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

### **`_M_swap()`**

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    const _Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### **`_M_transfer_from_if()`**

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if (
    const _Safe_sequence< _Sequence > & __from,
    _Predicate __pred) const
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

References `std::addressof()`, `__gnu_debug::_Safe_sequence_base::_M_const_iterators`, `__gnu_debug::_Safe_sequence_base::_M_g`, `__gnu_debug::_Safe_sequence_base::_M_iterators`, `__gnu_debug::_Safe_iterator_base::_M_next`, `__gnu_debug::_Safe_iterator_base::_M_sequence`, `__gnu_debug::_Safe_iterator_base::_M_version`, and `__gnu_debug::_Safe_sequence`.

## **5.155.3 Member Data Documentation**

### **`_M_const_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [mutable], [inherited]
```

The list of constant iterators that reference this container.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_unordered_container< _Container, _Key, _Compare, _Hash, _Alloc >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

### **`_M_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [mutable], [inherited]
```

The list of mutable iterators that reference this container.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_unordered_container< _Container, _Key, _Compare, _Hash, _Alloc >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

### **`_M_version`**

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Referenced by `_M_invalidate_all()`, `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator_base::_M_next`, `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=()`, `__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=()`, and `__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=()`.

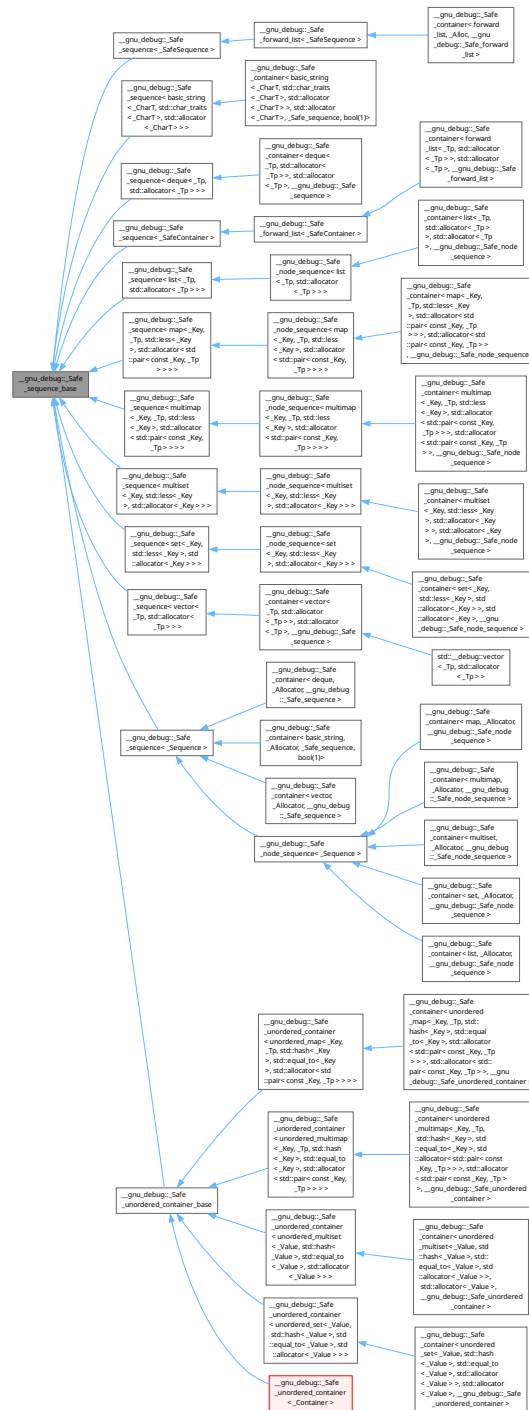
The documentation for this class was generated from the following files:

- `formatter.h`
- `safe_sequence.h`
- `safe_sequence.tcc`

## **5.156 `__gnu_debug::_Safe_sequence_base` Class Reference**

```
#include <safe_base.h>
```

Inheritance diagram for \_\_gnu\_debug::\_Safe\_sequence\_base:



## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- constexpr [\\_Safe\\_sequence\\_base](#) ([\\_Safe\\_sequence\\_base](#) &&\_\_seq) noexcept
- constexpr [\\_Safe\\_sequence\\_base](#) (const [\\_Safe\\_sequence\\_base](#) &) noexcept
- constexpr [~\\_Safe\\_sequence\\_base](#) ()
- void [\\_M\\_detach\\_all](#) () const
- void [\\_M\\_detach\\_singular](#) () const
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () const noexcept
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) () const
- void [\\_M\\_swap](#) (const [\\_Safe\\_sequence\\_base](#) &\_\_x) const noexcept

## Friends

- class [\\_Safe\\_iterator\\_base](#)

### 5.156.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.

The [\\_Safe\\_sequence\\_base](#) class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from [\\_Safe\\_sequence\\_base](#) publicly, so that safe iterators (which inherit [\\_Safe\\_iterator\\_base](#)) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

### 5.156.2 Constructor & Destructor Documentation

#### [~\\_Safe\\_sequence\\_base\(\)](#)

```
\_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base () [inline], [constexpr], [protected]
```

Notify all iterators that reference this sequence that the sequence is being destroyed.

References [\\_M\\_detach\\_all\(\)](#).

### 5.156.3 Member Function Documentation

#### [\\_M\\_detach\\_all\(\)](#)

```
void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_all () const [protected]
```

Detach all iterators, leaving them singular.

Referenced by [~\\_Safe\\_sequence\\_base\(\)](#).

#### [\\_M\\_detach\\_singular\(\)](#)

```
void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_singular () const [protected]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->[\\_M\\_version](#) == [\\_M\\_version](#).

#### [\\_M\\_get\\_mutex\(\)](#)

```
\_\_gnu\_cxx::\_\_mutex & \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_get\_mutex () const [protected], [noexcept]
```

For use in [\\_Safe\\_sequence](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container, \\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_local\\_if\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#).

**`_M_invalidate_all()`**

```
void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const [inline], [protected]
```

Invalidates all iterators.

References [\\_M\\_invalidate\\_all\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\\_all\(\)](#).

**`_M_revalidate_singular()`**

```
void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () const [protected]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**`_M_swap()`**

```
void __gnu_debug::Safe_sequence_base::_M_swap (
    const Safe_sequence_base & __x) const [protected], [noexcept]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.156.4 Member Data Documentation****`_M_const_iterators`**

```
Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators [mutable]
```

The list of constant iterators that reference this container.

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_if\(\)](#), and [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

**`_M_iterators`**

```
Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators [mutable]
```

The list of mutable iterators that reference this container.

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_if\(\)](#), and [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

**`_M_version`**

```
unsigned int __gnu_debug::Safe_sequence_base::_M_version [mutable]
```

The container version number. This number may never be 0.

Referenced by [\\_M\\_invalidate\\_all\(\)](#), [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), [\\_\\_gnu\\_debug::Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::Safe\\_local\\_iterator<\\_OtherIterator, UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::Safe\\_local\\_iterator<\\_OtherIterator, UContainer>::operator=\(\)](#).

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

**5.157 `__gnu_debug::Safe_unordered_container<_Container>` Class Template Reference**

```
#include <safe_unordered_container.h>
```



Inheritance diagram for `__gnu_debug::__Safe_unordered_container< _Container >`:



### Public Member Functions

- `void _M_invalidate_all ()`
- `template<typename _Predicate>`  
`void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Predicate>`  
`void _M_invalidate_local_if (_Predicate __pred)`

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all () const`
- `void _M_detach_singular () const`
- `__gnu_cxx::__mutex & _M_get_mutex () const noexcept`
- `void _M_invalidate_all () const`
- `void _M_invalidate_locals ()`
- `void _M_revalidate_singular () const`
- `void _M_swap (const _Safe_sequence_base &__x) const noexcept`
- `void _M_swap (const _Safe_unordered_container_base &__x) const noexcept`

### Static Protected Member Functions

- `template<typename _ExtractKey, typename _Source>`  
`static _UContMergeGuard< _Source, _UContInvalidatePred< _ExtractKey, _Source > > _S_uc_guard (_↵  
_ExtractKey, _Source &__src)`

- `template<typename _ExtractKey, typename _Source>`  
`static _UContMergeGuard<_Source, _UMContInvalidatePred<_ExtractKey, _Source>> _S_umc_guard (↔`  
`_ExtractKey, _Source &__src)`

### 5.157.1 Detailed Description

`template<typename _Container>`

`class __gnu_debug::__Safe_unordered_container<_Container>`

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.

The class template `_Safe_unordered_container` simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_↔ Safe_iterator` for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template `_Safe_local_iterator` for this container. Iterators will then be tracked automatically.

### 5.157.2 Member Function Documentation

#### `_M_detach_all()`

`void __gnu_debug::__Safe_unordered_container_base::_M_detach_all () const [protected], [inherited]`

Detach all iterators, leaving them singular.

Referenced by [~\\_Safe\\_unordered\\_container\\_base\(\)](#).

#### `_M_detach_singular()`

`void __gnu_debug::__Safe_sequence_base::_M_detach_singular () const [protected], [inherited]`

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### `_M_get_mutex()`

`__gnu_cxx::mutex & __gnu_debug::__Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]`

For use in `_Safe_sequence`.

Referenced by [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_local\\_if\(\)](#), and [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#).

#### `_M_invalidate_all()`

`void __gnu_debug::__Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]`

Invalidates all iterators.

References [\\_M\\_invalidate\\_all\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\\_all\(\)](#).

#### `_M_invalidate_if()`

`template<typename _Container>`

`template<typename _Predicate>`

`void __gnu_debug::__Safe_unordered_container<_Container>::_M_invalidate_if (`  
`_Predicate __pred)`

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

References [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_const\\_iterators](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_get\\_mutex\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_iterators](#).

### **`_M_invalidate_local_if()`**

```
template<typename _Container>
template<typename _Predicate>
void __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if (
    _Predicate __pred)
```

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal local iterators nested in the safe ones.

References [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container\\_base::\\_M\\_const\\_local\\_iterators](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::\\_M\\_get\\_mutex\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container\\_base::\\_M\\_local\\_iterators](#).

### **`_M_revalidate_singular()`**

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

### **`_M_swap()` [1/2]**

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    const _Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### **`_M_swap()` [2/2]**

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
    const _Safe_unordered_container_base & __x) const [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## **5.157.3 Member Data Documentation**

### **`_M_const_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [mutable], [inherited]
```

The list of constant iterators that reference this container.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence< \\_Sequence >::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container< \\_Container >::\\_M\\_invalidate\\_if\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence< \\_Sequence >::\\_M\\_transfer\\_from\\_if\(\)](#).

### **`_M_const_local_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [mutable],
[inherited]
```

The list of constant local iterators that reference this container.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container< \\_Container >::\\_M\\_invalidate\\_local\\_if\(\)](#).

### **`_M_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [mutable], [inherited]
```

The list of mutable iterators that reference this container.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_unordered_container<_Container>::_M_invalidate_if()` and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### `_M_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [mutable],
[inherited]
```

The list of mutable local iterators that reference this container.

Referenced by `__gnu_debug::_Safe_unordered_container<_Container>::_M_invalidate_local_if()`.

### `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Referenced by `_M_invalidate_all()`, `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()` and `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`.

The documentation for this class was generated from the following files:

- `safe_unordered_container.h`
- `safe_unordered_container.tcc`

## 5.158 `__gnu_debug::_Safe_unordered_container_base` Class Reference

```
#include <safe_unordered_base.h>
```

Inheritance diagram for `__gnu_debug::_Safe_unordered_container_base`:



## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `_Safe_unordered_container_base ( _Safe_unordered_container_base &&__x) noexcept`

- `_Safe_unordered_container_base` (const `_Safe_unordered_container_base` &) noexcept
- `~_Safe_unordered_container_base` () noexcept
- void `_M_detach_all` () const
- void `_M_detach_singular` () const
- `__gnu_cxx::__mutex` & `_M_get_mutex` () const noexcept
- void `_M_invalidate_all` () const
- void `_M_revalidate_singular` () const
- void `_M_swap` (const `_Safe_sequence_base` &\_\_x) const noexcept
- void `_M_swap` (const `_Safe_unordered_container_base` &\_\_x) const noexcept

## Friends

- class `_Safe_local_iterator_base`

### 5.158.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.

The `_Safe_unordered_container_base` class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from `_Safe_unordered_container_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

### 5.158.2 Constructor & Destructor Documentation

#### `~_Safe_unordered_container_base()`

```
__gnu_debug::__Safe_unordered_container_base::~~_Safe_unordered_container_base () [inline], [protected],
[noexcept]
```

Notify all iterators that reference this container that the container is being destroyed.

References `_M_detach_all()`.

### 5.158.3 Member Function Documentation

#### `_M_detach_all()`

```
void __gnu_debug::__Safe_unordered_container_base::_M_detach_all () const [protected]
```

Detach all iterators, leaving them singular.

Referenced by `~_Safe_unordered_container_base()`.

#### `_M_detach_singular()`

```
void __gnu_debug::__Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
```

Detach all singular iterators.

## Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

### **`_M_get_mutex()`**

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]
```

For use in `_Safe_sequence`.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_if\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_local\\_if\(\)](#).

### **`_M_invalidate_all()`**

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

References [\\_M\\_invalidate\\_all\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\\_all\(\)](#).

### **`_M_revalidate_singular()`**

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

### **`_M_swap()` [1/2]**

```
void __gnu_debug::_Safe_sequence_base::_M_swap (const _Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### **`_M_swap()` [2/2]**

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (const _Safe_unordered_container_base & __x) const [protected], [noexcept]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## **5.158.4 Member Data Documentation**

### **`_M_const_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [mutable], [inherited]
```

The list of constant iterators that reference this container.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_if\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

### **`_M_const_local_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [mutable]
```

The list of constant local iterators that reference this container.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_local\\_if\(\)](#).

### **`_M_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [mutable], [inherited]
```

The list of mutable iterators that reference this container.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_invalidate\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>::\\_M\\_invalidate\\_if\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

**`_M_local_iterators`**

`_Safe_iterator_base*` `__gnu_debug::_Safe_unordered_container_base::_M_local_iterators` [mutable]

The list of mutable local iterators that reference this container.

Referenced by `__gnu_debug::_Safe_unordered_container<_Container>::_M_invalidate_local_if()`.

**`_M_version`**

`unsigned int` `__gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Referenced by `_M_invalidate_all()`, `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>` and `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`.

The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)

## 5.159 `__gnu_debug::_Safe_vector<_SafeSequence, _BaseSequence>` Class Template Reference

```
#include <vector>
```

Inheritance diagram for `__gnu_debug::_Safe_vector<_SafeSequence, _BaseSequence>`:

**Protected Member Functions**

- `constexpr _Safe_vector` (`_Safe_vector` && `_x`) `noexcept`
- `constexpr _Safe_vector` (`const _Safe_vector` &) `noexcept`
- `constexpr _Safe_vector` (`size_type` `_n`) `noexcept`
- `bool` `_M_requires_reallocation` (`size_type` `_elements`) `const noexcept`



- constexpr void **\_M\_update\_guaranteed\_capacity** () noexcept
- constexpr [\\_Safe\\_vector](#) & **operator=** ([\\_Safe\\_vector](#) &&\_\_x) noexcept
- constexpr [\\_Safe\\_vector](#) & **operator=** (const [\\_Safe\\_vector](#) &) noexcept

#### Protected Attributes

- size\_type **\_M\_guaranteed\_capacity**

#### 5.159.1 Detailed Description

```
template<typename _SafeSequence, typename _BaseSequence>
class __gnu_debug:: _Safe_vector< _SafeSequence, _BaseSequence >
```

Base class for Debug Mode vector.

Adds information about the guaranteed capacity, which is useful for detecting code which relies on non-portable implementation details of the libstdc++ reallocation policy.

The documentation for this class was generated from the following file:

- [debug/vector](#)

#### 5.160 [\\_\\_gnu\\_parallel::\\_SamplingSorter](#)< [\\_\\_stable](#), [\\_RAIter](#), [\\_StrictWeakOrdering](#) > Struct Template Reference

```
#include <multiway_merge.h>
```

#### Public Member Functions

- void **operator()** ([\\_RAIter](#) \_\_first, [\\_RAIter](#) \_\_last, [\\_StrictWeakOrdering](#) \_\_comp)

#### 5.160.1 Detailed Description

```
template<bool __stable, class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel:: _SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >
```

Stable sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 5.161 [\\_\\_gnu\\_parallel::\\_SamplingSorter](#)< [false](#), [\\_RAIter](#), [\\_StrictWeakOrdering](#) > Struct Template Reference

```
#include <multiway_merge.h>
```

#### Public Member Functions

- void **operator()** ([\\_RAIter](#) \_\_first, [\\_RAIter](#) \_\_last, [\\_StrictWeakOrdering](#) \_\_comp)
- void **operator()** ([\\_RAIter](#) \_\_first, [\\_RAIter](#) \_\_last, [\\_StrictWeakOrdering](#) \_\_comp)

#### 5.161.1 Detailed Description

```
template<class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel:: _SamplingSorter< false, _RAIter, _StrictWeakOrdering >
```

Non-`__stable` sorting functor.

Used to reduce code instantiation in multiway\_merge\_sampling\_splitting.  
The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 5.162 std::\_\_detail::\_Scanner<\_CharT> Class Template Reference

```
#include <regex_scanner.h>
```

### Public Types

- typedef const [std::ctype](#)<\_CharT> **\_CtypeT**
- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef [std::basic\\_string](#)<\_CharT> **\_StringT**
- enum **\_TokenT** : unsigned {  
     **\_S\_token\_anch**, **\_S\_token\_ord\_char**, **\_S\_token\_oct\_num**, **\_S\_token\_hex\_num**,  
     **\_S\_token\_backref**, **\_S\_token\_subexpr\_begin**, **\_S\_token\_subexpr\_no\_group\_begin**, **\_S\_token\_subexpr\_lookahead\_begin**,  
     **\_S\_token\_subexpr\_end**, **\_S\_token\_bracket\_begin**, **\_S\_token\_bracket\_neg\_begin**, **\_S\_token\_bracket\_end**,  
     **\_S\_token\_interval\_begin**, **\_S\_token\_interval\_end**, **\_S\_token\_quoted\_class**, **\_S\_token\_char\_class\_name**,  
     **\_S\_token\_collsymbol**, **\_S\_token\_equiv\_class\_name**, **\_S\_token\_opt**, **\_S\_token\_or**,  
     **\_S\_token\_closure0**, **\_S\_token\_closure1**, **\_S\_token\_line\_begin**, **\_S\_token\_line\_end**,  
     **\_S\_token\_word\_bound**, **\_S\_token\_comma**, **\_S\_token\_dup\_count**, **\_S\_token\_eof**,  
     **\_S\_token\_bracket\_dash**, **\_S\_token\_unknown** }

### Public Member Functions

- **\_Scanner** (const \_CharT \*\_\_begin, const \_CharT \*\_\_end, [\\_FlagT](#) \_\_flags, [std::locale](#) \_\_loc)
- void **\_M\_advance** ()
- [\\_TokenT](#) **\_M\_get\_token** () const noexcept
- const [\\_StringT](#) & **\_M\_get\_value** () const noexcept

### Protected Types

- enum **\_StateT** { **\_S\_state\_normal**, **\_S\_state\_in\_brace**, **\_S\_state\_in\_bracket** }

### Protected Member Functions

- const char \* **\_M\_find\_escape** (char \_\_c)
- bool **\_M\_is\_awk** () const
- bool **\_M\_is\_basic** () const
- bool **\_M\_is\_ecma** () const
- bool **\_M\_is\_extended** () const
- bool **\_M\_is\_grep** () const

### Protected Attributes

- bool **\_M\_at\_bracket\_start**
- const [std::pair](#)< char, char > **\_M\_awk\_escape\_tbl** [11]
- const char \* **\_M\_basic\_spec\_char**
- const [std::pair](#)< char, char > **\_M\_ecma\_escape\_tbl** [8]
- const char \* **\_M\_ecma\_spec\_char**

- const [std::pair](#)< char, char > \* **\_M\_escape\_tbl**
- const char \* **\_M\_extended\_spec\_char**
- [\\_FlagT](#) **\_M\_flags**
- const char \* **\_M\_spec\_char**
- [\\_StateT](#) **\_M\_state**
- [\\_TokenT](#) **\_M\_token**
- const [std::pair](#)< char, [\\_TokenT](#) > **\_M\_token\_tbl** [9]

#### 5.162.1 Detailed Description

```
template<typename _CharT>
class std::__detail::_Scanner< _CharT >
```

Scans an input range for regex tokens.

The `_Scanner` class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

#### 5.162.2 Member Enumeration Documentation

##### `_TokenT`

```
enum std::__detail::_ScannerBase::_TokenT : unsigned [inherited]
```

Token types returned from the scanner.

The documentation for this class was generated from the following files:

- [regex\\_scanner.h](#)
- [regex\\_scanner.tcc](#)

### 5.163 `__gnu_debug::_Sequence_traits< _Sequence >` Struct Template Reference

```
#include <safe_iterator.h>
```

#### Public Types

- typedef `_Distance_traits< typename _Sequence::iterator >` **\_DistTraits**

#### Static Public Member Functions

- static `_DistTraits::__type` **\_S\_size** (const `_Sequence` &\_\_seq)

#### 5.163.1 Detailed Description

```
template<typename _Sequence>
struct __gnu_debug::_Sequence_traits< _Sequence >
```

Sequence traits giving the size of a container if possible.

The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

### 5.164 `__gnu_parallel::_Settings` Struct Reference

```
#include <settings.h>
```

**Static Public Member Functions**

- static const `_Settings` & `get` () throw ()
- static void `set` (`_Settings` &) throw ()

**Public Attributes**

- `_SequenceIndex` `accumulate_minimal_n`
- unsigned int `adjacent_difference_minimal_n`
- `_AlgorithmStrategy` `algorithm_strategy`
- unsigned int `cache_line_size`
- `_SequenceIndex` `count_minimal_n`
- `_SequenceIndex` `fill_minimal_n`
- `_FindAlgorithm` `find_algorithm`
- double `find_increasing_factor`
- `_SequenceIndex` `find_initial_block_size`
- `_SequenceIndex` `find_maximum_block_size`
- float `find_scale_factor`
- `_SequenceIndex` `find_sequential_search_size`
- `_SequenceIndex` `for_each_minimal_n`
- `_SequenceIndex` `generate_minimal_n`
- unsigned long long `L1_cache_size`
- unsigned long long `L2_cache_size`
- `_SequenceIndex` `max_element_minimal_n`
- `_SequenceIndex` `merge_minimal_n`
- unsigned int `merge_oversampling`
- `_SplittingAlgorithm` `merge_splitting`
- `_SequenceIndex` `min_element_minimal_n`
- `_MultiwayMergeAlgorithm` `multiway_merge_algorithm`
- int `multiway_merge_minimal_k`
- `_SequenceIndex` `multiway_merge_minimal_n`
- unsigned int `multiway_merge_oversampling`
- `_SplittingAlgorithm` `multiway_merge_splitting`
- `_SequenceIndex` `nth_element_minimal_n`
- `_SequenceIndex` `partial_sort_minimal_n`
- `_PartialSumAlgorithm` `partial_sum_algorithm`
- float `partial_sum_dilation`
- unsigned int `partial_sum_minimal_n`
- double `partition_chunk_share`
- `_SequenceIndex` `partition_chunk_size`
- `_SequenceIndex` `partition_minimal_n`
- `_SequenceIndex` `qsb_steals`
- unsigned int `random_shuffle_minimal_n`
- `_SequenceIndex` `replace_minimal_n`
- `_SequenceIndex` `search_minimal_n`
- `_SequenceIndex` `set_difference_minimal_n`
- `_SequenceIndex` `set_intersection_minimal_n`
- `_SequenceIndex` `set_symmetric_difference_minimal_n`
- `_SequenceIndex` `set_union_minimal_n`
- `_SortAlgorithm` `sort_algorithm`
- `_SequenceIndex` `sort_minimal_n`
- unsigned int `sort_mwms_oversampling`

- unsigned int [sort\\_qs\\_num\\_samples\\_preset](#)
- [\\_SequenceIndex](#) [sort\\_qsb\\_base\\_case\\_maximal\\_n](#)
- [\\_SplittingAlgorithm](#) [sort\\_splitting](#)
- unsigned int [TLB\\_size](#)
- [\\_SequenceIndex](#) [transform\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [unique\\_copy\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [workstealing\\_chunk\\_size](#)

### 5.164.1 Detailed Description

class `_Settings` Run-time settings for the parallel mode including all tunable parameters.

### 5.164.2 Member Function Documentation

#### `get()`

```
static const \_Settings & __gnu_parallel::_Settings::get () throw ( ) [static]
```

Get the global settings.

Referenced by [\\_\\_gnu\\_parallel::\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#), [\\_\\_gnu\\_parallel::\\_parallel\\_nth\\_element\(\)](#), [\\_\\_gnu\\_parallel::\\_parallel\\_partial\\_sum\(\)](#), [\\_\\_gnu\\_parallel::\\_parallel\\_partial\\_sum\\_linear\(\)](#), [\\_\\_gnu\\_parallel::\\_parallel\\_partition\(\)](#), [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [\\_\\_gnu\\_parallel::\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_parallel\\_sort\\_qs\\_conquer\(\)](#), [\\_\\_gnu\\_parallel::\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#), [\\_\\_gnu\\_parallel::\\_sequential\\_random\\_s](#), [\\_\\_gnu\\_parallel::multiway\\_merge\\_sampling\\_splitting\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_multiway\\_merge\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#).

#### `set()`

```
static void __gnu_parallel::_Settings::set ( \_Settings & ) throw ( ) [static]
```

Set the global settings.

References [accumulate\\_minimal\\_n](#), [adjacent\\_difference\\_minimal\\_n](#), [cache\\_line\\_size](#), [count\\_minimal\\_n](#), [fill\\_minimal\\_n](#), [find\\_increasing\\_factor](#), [find\\_initial\\_block\\_size](#), [find\\_maximum\\_block\\_size](#), [find\\_scale\\_factor](#), [find\\_sequential\\_search\\_size](#), [for\\_each\\_minimal\\_n](#), [generate\\_minimal\\_n](#), [L1\\_cache\\_size](#), [L2\\_cache\\_size](#), [max\\_element\\_minimal\\_n](#), [merge\\_minimal\\_n](#), [merge\\_oversampling](#), [min\\_element\\_minimal\\_n](#), [multiway\\_merge\\_minimal\\_k](#), [multiway\\_merge\\_minimal\\_n](#), [multiway\\_merge\\_oversampling](#), [nth\\_element\\_minimal\\_n](#), [partial\\_sort\\_minimal\\_n](#), [partial\\_sum\\_dilation](#), [partial\\_sum\\_minimal\\_n](#), [partition\\_chunk\\_share](#), [partition\\_chunk\\_size](#), [partition\\_minimal\\_n](#), [qsb\\_steals](#), [random\\_shuffle\\_minimal\\_n](#), [replace\\_minimal\\_n](#), [search\\_minimal\\_n](#), [set\(\)](#), [set\\_difference\\_minimal\\_n](#), [set\\_intersection\\_minimal\\_n](#), [set\\_symmetric\\_difference\\_minimal\\_n](#), [set\\_union\\_minimal\\_n](#), [sort\\_minimal\\_n](#), [sort\\_mwms\\_oversampling](#), [sort\\_qs\\_num\\_samples\\_preset](#), [sort\\_qsb\\_base\\_case\\_maximal\\_n](#), [TLB\\_size](#), [transform\\_minimal\\_n](#), and [unique\\_copy\\_minimal\\_n](#).

Referenced by [set\(\)](#).

### 5.164.3 Member Data Documentation

#### `accumulate_minimal_n`

```
\_SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n
```

Minimal input size for accumulate.

Referenced by [set\(\)](#).

#### `adjacent_difference_minimal_n`

```
unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n
```

Minimal input size for adjacent\_difference.

Referenced by [set\(\)](#).

**cache\_line\_size**

```
unsigned int __gnu_parallel::_Settings::cache_line_size
```

Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#), and [set\(\)](#).

**count\_minimal\_n**

```
\_SequenceIndex __gnu_parallel::_Settings::count_minimal_n
```

Minimal input size for count and count\_if.

Referenced by [set\(\)](#).

**fill\_minimal\_n**

```
\_SequenceIndex __gnu_parallel::_Settings::fill_minimal_n
```

Minimal input size for fill.

Referenced by [set\(\)](#).

**find\_increasing\_factor**

```
double __gnu_parallel::_Settings::find_increasing_factor
```

Block size increase factor for find.

Referenced by [set\(\)](#).

**find\_initial\_block\_size**

```
\_SequenceIndex __gnu_parallel::_Settings::find_initial_block_size
```

Initial block size for find.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), and [set\(\)](#).

**find\_maximum\_block\_size**

```
\_SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size
```

Maximal block size for find.

Referenced by [set\(\)](#).

**find\_scale\_factor**

```
float __gnu_parallel::_Settings::find_scale_factor
```

Block size scale-down factor with respect to current position.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), and [set\(\)](#).

**find\_sequential\_search\_size**

```
\_SequenceIndex __gnu_parallel::_Settings::find_sequential_search_size
```

Start with looking for this many elements sequentially, for find.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), and [set\(\)](#).

**for\_each\_minimal\_n**

```
\_SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n
```

Minimal input size for for\_each.

Referenced by [set\(\)](#).

**generate\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::generate_minimal_n`

Minimal input size for generate.

Referenced by [set\(\)](#).

**L1\_cache\_size**

`unsigned long long __gnu_parallel::_Settings::L1_cache_size`

size of the L1 cache in bytes (underestimation).

Referenced by [set\(\)](#).

**L2\_cache\_size**

`unsigned long long __gnu_parallel::_Settings::L2_cache_size`

size of the L2 cache in bytes (underestimation).

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_sequential\\_random\\_shuffle\(\)](#), and [set\(\)](#).

**max\_element\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::max_element_minimal_n`

Minimal input size for max\_element.

Referenced by [set\(\)](#).

**merge\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::merge_minimal_n`

Minimal input size for merge.

Referenced by [set\(\)](#).

**merge\_oversampling**

`unsigned int __gnu_parallel::_Settings::merge_oversampling`

Oversampling factor for merge.

Referenced by [\\_\\_gnu\\_parallel::multiway\\_merge\\_sampling\\_splitting\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_multiway\\_merge\(\)](#), and [set\(\)](#).

**min\_element\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::min_element_minimal_n`

Minimal input size for min\_element.

Referenced by [set\(\)](#).

**multiway\_merge\_minimal\_k**

`int __gnu_parallel::_Settings::multiway_merge_minimal_k`

Oversampling factor for multiway\_merge.

Referenced by [set\(\)](#).

**multiway\_merge\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::multiway_merge_minimal_n`

Minimal input size for multiway\_merge.

Referenced by [set\(\)](#).

**multiway\_merge\_oversampling**

`unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling`

Oversampling factor for `multiway_merge`.

Referenced by [set\(\)](#).

**nth\_element\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::nth_element_minimal_n`

Minimal input size for `nth_element`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_nth\\_element\(\)](#), and [set\(\)](#).

**partial\_sort\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::partial_sort_minimal_n`

Minimal input size for `partial_sort`.

Referenced by [set\(\)](#).

**partial\_sum\_dilation**

`float __gnu_parallel::_Settings::partial_sum_dilation`

Ratio for `partial_sum`. Assume "sum and write result" to be this factor slower than just "sum".

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partial\\_sum\\_linear\(\)](#), and [set\(\)](#).

**partial\_sum\_minimal\_n**

`unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n`

Minimal input size for `partial_sum`.

Referenced by [set\(\)](#).

**partition\_chunk\_share**

`double __gnu_parallel::_Settings::partition_chunk_share`

Chunk size for partition, relative to input size. If  $> 0.0$ , this value overrides `partition_chunk_size`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partition\(\)](#), and [set\(\)](#).

**partition\_chunk\_size**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::partition_chunk_size`

Chunk size for partition.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partition\(\)](#), and [set\(\)](#).

**partition\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::partition_minimal_n`

Minimal input size for partition.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_nth\\_element\(\)](#), and [set\(\)](#).

**qsb\_steals**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::qsb_steals`

The number of stolen ranges in load-balanced quicksort.

Referenced by [set\(\)](#).



**random\_shuffle\_minimal\_n**

unsigned int \_\_gnu\_parallel::\_Settings::random\_shuffle\_minimal\_n  
Minimal input size for random\_shuffle.  
Referenced by [set\(\)](#).

**replace\_minimal\_n**

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::replace\_minimal\_n  
Minimal input size for replace and replace\_if.  
Referenced by [set\(\)](#).

**search\_minimal\_n**

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::search\_minimal\_n  
Minimal input size for search and search\_n.  
Referenced by [set\(\)](#).

**set\_difference\_minimal\_n**

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::set\_difference\_minimal\_n  
Minimal input size for set\_difference.  
Referenced by [set\(\)](#).

**set\_intersection\_minimal\_n**

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::set\_intersection\_minimal\_n  
Minimal input size for set\_intersection.  
Referenced by [set\(\)](#).

**set\_symmetric\_difference\_minimal\_n**

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::set\_symmetric\_difference\_minimal\_n  
Minimal input size for set\_symmetric\_difference.  
Referenced by [set\(\)](#).

**set\_union\_minimal\_n**

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::set\_union\_minimal\_n  
Minimal input size for set\_union.  
Referenced by [set\(\)](#).

**sort\_minimal\_n**

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::sort\_minimal\_n  
Minimal input size for parallel sorting.  
Referenced by [set\(\)](#).

**sort\_mwms\_oversampling**

unsigned int \_\_gnu\_parallel::\_Settings::sort\_mwms\_oversampling  
Oversampling factor for parallel std::sort (MWMS).  
Referenced by [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#), and [set\(\)](#).

**sort\_qs\_num\_samples\_preset**

`unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Referenced by [set\(\)](#).

**sort\_qsb\_base\_case\_maximal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`

Maximal subsequence `__length` to switch to unbalanced `__base` case. Applies to `std::sort` with dynamically load-balanced quicksort.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#), and [set\(\)](#).

**TLB\_size**

`unsigned int __gnu_parallel::_Settings::TLB_size`

size of the Translation Lookaside Buffer (underestimation).

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_sequential\\_random\\_shuffle\(\)](#), and [set\(\)](#).

**transform\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::transform_minimal_n`

Minimal input size for parallel `std::transform`.

Referenced by [set\(\)](#).

**unique\_copy\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::unique_copy_minimal_n`

Minimal input size for `unique_copy`.

Referenced by [set\(\)](#).

The documentation for this struct was generated from the following file:

- [settings.h](#)

**5.165 `std::_Sp_ebo_helper<_Nm, _Tp, false >` Struct Template Reference**

```
#include <shared_ptr_base.h>
```

**Public Member Functions**

- `_Sp_ebo_helper` (`_Tp &&__tp`)
- `_Sp_ebo_helper` (`const _Tp &__tp`)

**Static Public Member Functions**

- `static _Tp &_S_get` (`_Sp_ebo_helper &__eboh`)

**5.165.1 Detailed Description**

`template<int _Nm, typename _Tp>`

`struct std::_Sp_ebo_helper<_Nm, _Tp, false >`

Specialization not using EBO.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

### 5.166 `std::_Sp_ebo_helper<_Nm, _Tp, true >` Struct Template Reference

```
#include <shared_ptr_base.h>
```

#### Public Member Functions

- `_Sp_ebo_helper` (`_Tp &&__tp`)
- `_Sp_ebo_helper` (`const _Tp &__tp`)

#### Static Public Member Functions

- `static _Tp &_S_get` (`_Sp_ebo_helper &__eboh`)

##### 5.166.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::_Sp_ebo_helper<_Nm, _Tp, true >
```

Specialization using EBO.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

### 5.167 `__gnu_parallel::SplitConsistently<__exact, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference

```
#include <multiway_mergesort.h>
```

#### 5.167.1 Detailed Description

```
template<bool __exact, typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::SplitConsistently<__exact, _RAIter, _Compare, _SortingPlacesIterator >
```

Split consistently.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

### 5.168 `__gnu_parallel::SplitConsistently<false, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference

```
#include <multiway_mergesort.h>
```

#### Public Member Functions

- `void operator()` (`const _ThreadIndex __iam, _PMWSSortingData<_RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits<_RAIter >::difference_type __num_samples`) `const`

##### 5.168.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::SplitConsistently<false, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by sampling.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 5.169 `__gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

```
#include <multiway_mergesort.h>
```

### Public Member Functions

- void **operator()** (const [\\_ThreadIndex](#) \_\_iam, [\\_PMWMSortingData](#)< \_RAIter > \*\_\_sd, \_Compare &\_\_comp, const typename [std::iterator\\_traits](#)< \_RAIter >::difference\_type \_\_num\_samples) const

#### 5.169.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>  
struct __gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 5.170 `std::__detail::StateSeq< _TraitsT > Class` Template Reference

```
#include <regex_automaton.h>
```

### Public Types

- typedef `_NFA< _TraitsT > _RegexT`

### Public Member Functions

- `_StateSeq` (`_RegexT &__nfa, _StateIdT __s`)
- `_StateSeq` (`_RegexT &__nfa, _StateIdT __s, _StateIdT __end`)
- void `_M_append` (`_StateIdT __id`)
- void `_M_append` (const [\\_StateSeq](#) &\_\_s)
- `_StateSeq` `_M_clone` ()

### Public Attributes

- `_StateIdT` `_M_end`
- `_RegexT` & `_M_nfa`
- `_StateIdT` `_M_start`

#### 5.170.1 Detailed Description

```
template<typename _TraitsT>  
class std::__detail::StateSeq< _TraitsT >
```

Describes a sequence of one or more `_State`, its current start and end(s). This structure contains fragments of an NFA during construction.

The documentation for this class was generated from the following files:

- [regex\\_automaton.h](#)
- [regex\\_automaton.tcc](#)

## 5.171 `__gnu_cxx::Std_pointer_impl< _Tp > Class` Template Reference

```
#include <pointer.h>
```

## Public Types

- typedef `_Tp` **element\_type**

## Public Member Functions

- `_Tp * get () const`
- `bool operator< (const \_Std\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Std\_pointer\_impl &__rarg) const`
- `void set (element_type *__arg)`

### 5.171.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Std_pointer_impl<_Tp>
```

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==()` to support pointer comparison.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.172 `__gnu_cxx::Temporary_buffer<_ForwardIterator, _Tp>` Class Template Reference

```
#include <stl_tempbuf.h>
```

Inheritance diagram for `__gnu_cxx::Temporary_buffer<_ForwardIterator, _Tp>`:



## Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `_Temporary_buffer` (`_ForwardIterator __seed`, `size_type __original_len`)
- `size_type` `_M_requested_size` () const
- iterator `begin` ()
- iterator `end` ()
- `size_type` `size` () const

## Protected Attributes

- struct `std::__Temporary_buffer::__Impl` `_M_impl`
- `size_type` `_M_original_len`

### 5.172.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp>
class __gnu_cxx::__Temporary_buffer<_ForwardIterator, _Tp>
```

This class is used in two places: `stl_algo.h` and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

### 5.172.2 Constructor & Destructor Documentation

#### `_Temporary_buffer()`

```
template<typename _ForwardIterator, typename _Tp>
std::__Temporary_buffer<_ForwardIterator, _Tp>::__Temporary_buffer (
    _ForwardIterator __seed,
    size_type __original_len)
```

Constructs a temporary buffer of a size somewhere between zero and the given length.

### 5.172.3 Member Function Documentation

#### `_M_requested_size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::__Temporary_buffer<_ForwardIterator, _Tp>::_M_requested_size () const [inline]
```

Returns the size requested by the constructor; may be `>size()`.

#### `begin()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::__Temporary_buffer<_ForwardIterator, _Tp>::begin () [inline]
```

As per Table mumble.

#### `end()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::__Temporary_buffer<_ForwardIterator, _Tp>::end () [inline]
```

As per Table mumble.

#### `size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::__Temporary_buffer<_ForwardIterator, _Tp>::size () const [inline]
```

As per Table mumble.

The documentation for this class was generated from the following file:

- [stl\\_tempbuf.h](#)

### 5.173 std::\_Temporary\_buffer<\_ForwardIterator, \_Tp> Class Template Reference

```
#include <stl_tempbuf.h>
```

Inheritance diagram for std::\_Temporary\_buffer<\_ForwardIterator, \_Tp>:



#### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- [\\_Temporary\\_buffer](#) (\_ForwardIterator \_\_seed, size\_type \_\_original\_len)
- size\_type [\\_M\\_requested\\_size](#) () const
- iterator [begin](#) ()
- iterator [end](#) ()
- size\_type [size](#) () const

#### Protected Attributes

- struct std::\_Temporary\_buffer::\_Impl **\_M\_impl**
- size\_type **\_M\_original\_len**

#### 5.173.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp>
class std::_Temporary_buffer<_ForwardIterator, _Tp>
```

This class is used in two places: `stl_algo.h` and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

### 5.173.2 Constructor & Destructor Documentation

#### `_Temporary_buffer()`

```
template<typename _ForwardIterator, typename _Tp>
std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer (
    _ForwardIterator __seed,
    size_type __original_len)
```

Constructs a temporary buffer of a size somewhere between zero and the given length.

References [begin\(\)](#), and [end\(\)](#).

### 5.173.3 Member Function Documentation

#### `_M_requested_size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const [inline]
```

Returns the size requested by the constructor; may be `>size()`.

#### `begin()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin () [inline]
```

As per Table mumble.

Referenced by [\\_Temporary\\_buffer\(\)](#).

#### `end()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end () [inline]
```

As per Table mumble.

Referenced by [\\_Temporary\\_buffer\(\)](#).

#### `size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const [inline]
```

As per Table mumble.

The documentation for this class was generated from the following file:

- [stl\\_tempbuf.h](#)

## 5.174 `__gnu_cxx::Unqualified_type<_Tp>` Struct Template Reference

```
#include <pointer.h>
```

### Public Types

- `typedef _Tp type`

### 5.174.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::Unqualified_type< _Tp >
```

This structure accommodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_type` is still `T`.

The documentation for this struct was generated from the following file:



- [pointer.h](#)

### 5.175 std::\_Vector\_base< \_Tp, \_Alloc > Struct Template Reference

```
#include <stl_vector.h>
```

Inheritance diagram for std::\_Vector\_base< \_Tp, \_Alloc >:



**Public Types**

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)<\_Alloc>::template rebind<\_Tp>::other **\_Tp\_alloc\_type**
- typedef \_Alloc **allocator\_type**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)<\_Tp\_alloc\_type>::pointer **pointer**

**Public Member Functions**

- constexpr **\_Vector\_base** (\_Tp\_alloc\_type &&\_\_a) noexcept
- **\_Vector\_base** (\_Vector\_base &&)=default
- constexpr **\_Vector\_base** (\_Vector\_base &&\_\_x, const allocator\_type &\_\_a)
- constexpr **\_Vector\_base** (const allocator\_type &\_\_a) noexcept
- constexpr **\_Vector\_base** (const allocator\_type &\_\_a, [\\_Vector\\_base](#) &&\_\_x)
- constexpr **\_Vector\_base** (size\_t \_\_n)
- constexpr **\_Vector\_base** (size\_t \_\_n, const allocator\_type &\_\_a)
- constexpr pointer **\_M\_allocate** (size\_t \_\_n)
- constexpr void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- constexpr const \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- constexpr \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- constexpr allocator\_type **get\_allocator** () const noexcept

**Public Attributes**

- [\\_Vector\\_impl](#) **\_M\_impl**

**Protected Member Functions**

- constexpr void **\_M\_create\_storage** (size\_t \_\_n)

**5.175.1 Detailed Description**

template<typename \_Tp, typename \_Alloc>

struct std::\_Vector\_base<\_Tp, \_Alloc>

See bits/stl\_deque.h's \_Deque\_base for an explanation.

The documentation for this struct was generated from the following file:

- [stl\\_vector.h](#)

**5.176 std::add\_const<\_Tp> Struct Template Reference**

```
#include <type_traits>
```

**Public Types**

- using **type**

**5.176.1 Detailed Description**

template<typename \_Tp>

struct std::add\_const<\_Tp>

add\_const

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.177 `std::add_cv<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### Public Types

- using `type`

#### 5.177.1 Detailed Description

```
template<typename _Tp>  
struct std::add_cv<_Tp>
```

`add_cv`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.178 `std::add_lvalue_reference<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### Public Types

- using `type`

#### 5.178.1 Detailed Description

```
template<typename _Tp>  
struct std::add_lvalue_reference<_Tp>
```

`add_lvalue_reference`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.179 `std::add_pointer<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### Public Types

- using `type`

#### 5.179.1 Detailed Description

```
template<typename _Tp>  
struct std::add_pointer<_Tp>
```

`add_pointer`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.180 `std::add_rvalue_reference<_Tp>` Struct Template Reference

```
#include <type_traits>
```

**Public Types**

- using `type`

**5.180.1 Detailed Description**

```
template<typename _Tp>
struct std::add_rvalue_reference<_Tp>
```

`add_rvalue_reference`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.181 `std::add_volatile<_Tp>` Struct Template Reference**

```
#include <type_traits>
```

**Public Types**

- using `type`

**5.181.1 Detailed Description**

```
template<typename _Tp>
struct std::add_volatile<_Tp>
```

`add_volatile`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.182 `std::adopt_lock_t` Struct Reference**

```
#include <std_mutex.h>
```

**5.182.1 Detailed Description**

Assume the calling thread has already obtained mutex ownership and manage it.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

**5.183 `std::aligned_storage<_Len, _Align>` Struct Template Reference**

```
#include <type_traits>
```

**5.183.1 Detailed Description**

```
template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
struct std::aligned_storage<_Len, _Align>
```

Aligned storage.

The member typedef `type` is be a POD type suitable for use as uninitialized storage for any object whose size is at most `_Len` and whose alignment is a divisor of `_Align`.

It is important to use the nested `type` as uninitialized storage, not the `std::aligned_storage` type itself which is an empty class with 1-byte alignment. So this is correct:

```
typename std::aligned_storage<sizeof(X), alignof(X)>::type m_xobj;
```

This is wrong:

```
std::aligned_storage<sizeof(X), alignof(X)> m_xobj;
```

In C++14 and later `std::aligned_storage_t<sizeof(X), alignof(X)>` can be used to refer to the type member typedef.

The default value of `_Align` is supposed to be the most stringent fundamental alignment requirement for any C++ object type whose size is no greater than `_Len` (see [basic.align] in the C++ standard).

**Deprecated** Deprecated in C++23. Uses can be replaced by an array `std::byte[_Len]` declared with `alignas(_Align)`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.184 `std::aligned_union<_Len, _Types >` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using [type](#)

### Static Public Attributes

- static const size\_t [alignment\\_value](#)

#### 5.184.1 Detailed Description

```
template<size_t _Len, typename... _Types>
```

```
struct std::aligned_union<_Len, _Types >
```

Provide aligned storage for types.

[meta.trans.other]

Provides aligned storage for any of the provided types of at least size `_Len`.

See also

`aligned_storage`

**Deprecated** Deprecated in C++23.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.185 `std::alignment_of<_Tp >` Struct Template Reference

```
#include <type_traits>
```

Inheritance diagram for `std::alignment_of<_Tp>`:



### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr `std::size_t` **value**

#### 5.185.1 Detailed Description

```
template<typename _Tp>
struct std::alignment_of<_Tp>
```

`alignment_of`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.186 `__gnu_cxx::allocator<_Tp>` Class Template Reference

```
#include <memory>
```

Inheritance diagram for `__gnu_cxx::allocator<_Tp>`:



### Public Types

- typedef ptrdiff\_t **difference\_type**
- using **is\_always\_equal**
- using **propagate\_on\_container\_move\_assignment**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **allocator** (const [allocator](#) &\_\_a) noexcept
- template<typename \_Tp1>  
constexpr **allocator** (const [allocator](#)<\_Tp1> &) noexcept
- constexpr \_Tp \* **allocate** (size\_t \_\_n)
- \_Tp \* **allocate** (size\_type \_\_n, const void \* \_\_p, size\_t \_\_n)
- constexpr void **deallocate** (\_Tp \* \_\_p, size\_t \_\_n)
- [allocator](#) & **operator=** (const [allocator](#) &)=default

### Friends

- constexpr bool **operator==** (const [allocator](#) &, const [allocator](#) &) noexcept

### Related Symbols

(Note that these are not member symbols.)

- template<typename \_T1, typename \_T2>  
constexpr bool **operator==** (const [allocator](#)<\_T1> &, const [allocator](#)<\_T2> &) noexcept

#### 5.186.1 Detailed Description

**template<typename \_Tp>**  
**class** `__gnu_cxx::allocator<_Tp>`

The *standard* allocator, as per C++03 [20.4.1].

See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator> for further details.

## Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

The documentation for this class was generated from the following file:

- [allocator.h](#)

5.187 `std::allocator<_Tp>` Class Template Reference

```
#include <memory>
```

Inheritance diagram for `std::allocator<_Tp>`:



## Public Types

- typedef `ptrdiff_t` **difference\_type**
- using **is\_always\_equal**
- using **propagate\_on\_container\_move\_assignment**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **allocator** (const [allocator](#) &\_\_a) noexcept
- template<typename `_Tp1`>  
constexpr **allocator** (const [allocator](#)< `_Tp1` > &) noexcept
- `_Tp` \* **allocate** (size\_type \_\_n, const void \* \_\_p) noexcept
- constexpr `_Tp` \* **allocate** (size\_t \_\_n)
- constexpr void **deallocate** (`_Tp` \* \_\_p, size\_t \_\_n)
- [allocator](#) & **operator=** (const [allocator](#) &)=default

## Friends

- constexpr bool **operator==** (const [allocator](#) &, const [allocator](#) &) noexcept

## Related Symbols

(Note that these are not member symbols.)

- template<typename `_T1`, typename `_T2`>  
constexpr bool **operator==** (const [allocator](#)< `_T1` > &, const [allocator](#)< `_T2` > &) noexcept



### 5.187.1 Detailed Description

```
template<typename _Tp>
class std::allocator<_Tp>
```

The *standard* allocator, as per C++03 [20.4.1].

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.↵](https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator)  
[memory.allocator](#) for further details.

#### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 5.188 `std::allocator_traits<_Alloc>` Struct Template Reference

```
#include <memory>
```

Inheritance diagram for std::allocator\_traits<\_Alloc>:



## Public Types

- typedef `_Alloc` `allocator_type`
- using `const_pointer`

- using [const\\_void\\_pointer](#)
- using [difference\\_type](#)
- using [is\\_always\\_equal](#)
- using [pointer](#)
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_swap](#)
- template<typename \_Tp>  
using [rebind\\_alloc](#)
- template<typename \_Tp>  
using [rebind\\_traits](#)
- using [size\\_type](#)
- typedef \_Alloc::value\_type [value\\_type](#)
- using [void\\_pointer](#)

### Static Public Member Functions

- static constexpr [pointer allocate](#) (\_Alloc &\_\_a, [size\\_type](#) \_\_n)
- static constexpr [pointer allocate](#) (\_Alloc &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#) \_\_hint)
- template<typename \_Tp, typename... \_Args>  
requires [\\_\\_can\\_construct](#)<\_Alloc, \_Tp, \_Args...>  
static constexpr void [construct](#) (\_Alloc &\_\_a, \_Tp \*\_\_p, \_Args &&... \_\_args) noexcept([\\_S\\_nothrow\\_construct](#)<\_Tp, \_Args... >())
- static constexpr void [deallocate](#) (\_Alloc &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- template<typename \_Tp>  
static constexpr void [destroy](#) (\_Alloc &\_\_a, \_Tp \*\_\_p) noexcept([\\_S\\_nothrow\\_destroy](#)<\_Tp >())
- static constexpr [size\\_type max\\_size](#) (const \_Alloc &\_\_a) noexcept
- static constexpr \_Alloc [select\\_on\\_container\\_copy\\_construction](#) (const \_Alloc &\_\_rhs)

#### 5.188.1 Detailed Description

**template<typename \_Alloc>**  
**struct std::allocator\_traits<\_Alloc >**

Uniform interface to all allocator types.

Since

C++11

#### 5.188.2 Member Typedef Documentation

##### **allocator\_type**

```
template<typename _Alloc>
typedef _Alloc std::allocator\_traits<_Alloc >::allocator_type
```

The allocator type.

##### **const\_pointer**

```
template<typename _Alloc>
using std::allocator\_traits<_Alloc >::const_pointer
```

The allocator's const pointer type.

[Alloc::const\\_pointer](#) if that type exists, otherwise [pointer\\_traits](#)<[pointer](#)>::rebind<const value\_type>

**const\_void\_pointer**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer
```

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

**difference\_type**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::difference_type
```

The allocator's difference type.

`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

**is\_always\_equal**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal
```

Whether all instances of the allocator type compare equal.

`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

**pointer**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::pointer
```

The allocator's pointer type.

`Alloc::pointer` if that type exists, otherwise `value_type*`

**propagate\_on\_container\_copy\_assignment**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

**propagate\_on\_container\_move\_assignment**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

**propagate\_on\_container\_swap**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap
```

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

**size\_type**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::size_type
```

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

### value\_type

```
template<typename _Alloc>
typedef _Alloc::value_type std::allocator_traits< _Alloc >::value_type
The allocated type.
```

### void\_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer
The allocator's void pointer type.
Alloc::void_pointer if that type exists, otherwise pointer_traits<pointer>::rebind<void>
```

## 5.188.3 Member Function Documentation

### allocate() [1/2]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
    _Alloc & __a,
    size_type __n) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

#### Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

### allocate() [2/2]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
    _Alloc & __a,
    size_type __n,
    const_void_pointer __hint) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

#### Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

#### Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

**construct()**

```
template<typename _Alloc>
template<typename _Tp, typename... _Args>
requires __can_construct<_Alloc, _Tp, _Args...>
static constexpr void std::allocator_traits<_Alloc>::construct (
    _Alloc & __a,
    _Tp * __p,
    _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`

**Parameters**

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

**deallocate()**

```
template<typename _Alloc>
static constexpr void std::allocator_traits<_Alloc>::deallocate (
    _Alloc & __a,
    pointer __p,
    size_type __n) [inline], [static], [constexpr]
```

Deallocate memory.

**Parameters**

|                        |                                                |
|------------------------|------------------------------------------------|
| <code>↔<br/>__a</code> | An allocator.                                  |
| <code>↔<br/>__p</code> | Pointer to the memory to deallocate.           |
| <code>↔<br/>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

**destroy()**

```
template<typename _Alloc>
template<typename _Tp>
static constexpr void std::allocator_traits<_Alloc>::destroy (
    _Alloc & __a,
    _Tp * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Tp`.

**Parameters**

|                        |               |
|------------------------|---------------|
| <code>↔<br/>__a</code> | An allocator. |
|------------------------|---------------|

|                  |                                  |
|------------------|----------------------------------|
| <code>__p</code> | Pointer to the object to destroy |
|------------------|----------------------------------|

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

### **max\_size()**

```
template<typename _Alloc>
static constexpr size_type std::allocator_traits< _Alloc >::max_size (
    const _Alloc & __a) [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

#### **Parameters**

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

#### **Returns**

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

### **select\_on\_container\_copy\_construction()**

```
template<typename _Alloc>
static constexpr _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (
    const _Alloc & __rhs) [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

#### **Parameters**

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

#### **Returns**

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

## **5.189 std::allocator\_traits< allocator< \_Tp > > Struct Template Reference**

```
#include <memory>
```

**Public Types**

- typedef [allocator< \\_Tp >](#) [allocator\\_type](#)
- using [allocator\\_type](#)
- using [const\\_pointer](#)
- using [const\\_pointer](#)
- using [const\\_void\\_pointer](#)
- using [const\\_void\\_pointer](#)
- using [difference\\_type](#)
- using [difference\\_type](#)
- using [is\\_always\\_equal](#)
- using [is\\_always\\_equal](#)
- using [pointer](#)
- using [pointer](#)
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_swap](#)
- using [propagate\\_on\\_container\\_swap](#)
- using **rebind\_alloc**
- template<typename \_Up>  
using **rebind\_alloc**
- using **rebind\_traits**
- template<typename \_Up>  
using **rebind\_traits**
- using [size\\_type](#)
- using [size\\_type](#)
- typedef [allocator< \\_Tp >::value\\_type](#) [value\\_type](#)
- using [value\\_type](#)
- using [void\\_pointer](#)
- using [void\\_pointer](#)

**Static Public Member Functions**

- static constexpr [pointer](#) [allocate](#) ([allocator< \\_Tp >](#) &\_\_a, [size\\_type](#) \_\_n)
- static constexpr [pointer](#) [allocate](#) ([allocator< \\_Tp >](#) &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#) \_\_hint)
- static constexpr [pointer](#) [allocate](#) ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n)
- static constexpr [pointer](#) [allocate](#) ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#) \_\_hint)
- static constexpr void [construct](#) ([allocator< \\_Tp >](#) &\_\_a, [\\_Tp \\*](#)\_\_p, [\\_Args](#) &&... \_\_args) noexcept([\\_S\\_nothrow\\_](#)↔[\\_construct< \\_Tp, \\_Args... >\(\)](#))
- template<typename \_Up, typename... \_Args>  
static constexpr void [construct](#) ([allocator\\_type](#) &\_\_a, [\\_Up \\*](#)\_\_p, [\\_Args](#) &&... \_\_args) noexcept([\\_is\\_nothrow\\_](#)↔[new\\_constructible< \\_Up, \\_Args... >\(\)](#))
- static constexpr void [deallocate](#) ([allocator< \\_Tp >](#) &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- static constexpr void [deallocate](#) ([allocator\\_type](#) &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- static constexpr void [destroy](#) ([allocator< \\_Tp >](#) &\_\_a, [\\_Tp \\*](#)\_\_p) noexcept([\\_S\\_nothrow\\_destroy< \\_Tp >\(\)](#))
- template<typename \_Up>  
static constexpr void [destroy](#) ([allocator\\_type](#) &\_\_a, [\\_Up \\*](#)\_\_p) noexcept([is\\_nothrow\\_destructible< \\_Up >::value](#))
- static constexpr [size\\_type](#) [max\\_size](#) (const [allocator< \\_Tp >](#) &\_\_a) noexcept
- static constexpr [size\\_type](#) [max\\_size](#) (const [allocator\\_type](#) &\_\_a) noexcept
- static constexpr [allocator< \\_Tp >](#) [select\\_on\\_container\\_copy\\_construction](#) (const [allocator< \\_Tp >](#) &\_\_rhs)
- static constexpr [allocator\\_type](#) [select\\_on\\_container\\_copy\\_construction](#) (const [allocator\\_type](#) &\_\_rhs)



### 5.189.1 Detailed Description

```
template<typename _Tp>
struct std::allocator_traits< allocator< _Tp > >
```

Partial specialization for `std::allocator`

Since

C++11

See also

`std::allocator_traits`

### 5.189.2 Member Typedef Documentation

**allocator\_type** [1/2]

```
typedef allocator< _Tp > std::allocator_traits< allocator< _Tp > >::allocator_type
```

The allocator type.

**allocator\_type** [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::allocator_type
```

The allocator type.

**const\_pointer** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::const_pointer
```

The allocator's const pointer type.  
`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

**const\_pointer** [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::const_pointer
```

The allocator's const pointer type.

**const\_void\_pointer** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::const_void_pointer
```

The allocator's const void pointer type.  
`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

**const\_void\_pointer** [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::const_void_pointer
```

The allocator's const void pointer type.

**difference\_type** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::difference_type
```

The allocator's difference type.

`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

**difference\_type** [2/2]

```
template<typename _Tp>
```

```
using std::allocator_traits< allocator< _Tp > >::difference_type
```

The allocator's difference type.

**is\_always\_equal** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

**is\_always\_equal** [2/2]

```
template<typename _Tp>
```

```
using std::allocator_traits< allocator< _Tp > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

**pointer** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::pointer
```

The allocator's pointer type.

`Alloc::pointer` if that type exists, otherwise `value_type*`

**pointer** [2/2]

```
template<typename _Tp>
```

```
using std::allocator_traits< allocator< _Tp > >::pointer
```

The allocator's pointer type.

**propagate\_on\_container\_copy\_assignment** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

**propagate\_on\_container\_copy\_assignment** [2/2]

```
template<typename _Tp>
```

```
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.

**propagate\_on\_container\_move\_assignment** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

**propagate\_on\_container\_move\_assignment** [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.

**propagate\_on\_container\_swap** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_swap
```

How the allocator is propagated on swap.  
Alloc::propagate\_on\_container\_swap if that type exists, otherwise false\_type

**propagate\_on\_container\_swap** [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_swap
```

How the allocator is propagated on swap.

**size\_type** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::size_type
```

The allocator's size type.  
Alloc::size\_type if that type exists, otherwise make\_unsigned<difference\_type>::type

**size\_type** [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::size_type
```

The allocator's size type.

**value\_type** [1/2]

```
typedef allocator< _Tp >::value_type std::allocator_traits< allocator< _Tp > >::value_type
```

The allocated type.

**value\_type** [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::value_type
```

The allocated type.

**void\_pointer** [1/2]

```
using std::allocator_traits< allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.  
Alloc::void\_pointer if that type exists, otherwise pointer\_traits<pointer>::rebind<void>

**void\_pointer** [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.

### 5.189.3 Member Function Documentation

#### allocate() [1/4]

```
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
    allocator< _Tp > & __a,
    size_type __n) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

##### Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

#### allocate() [2/4]

```
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
    allocator< _Tp > & __a,
    size_type __n,
    const_void_pointer __hint) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

##### Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

##### Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

#### allocate() [3/4]

```
template<typename _Tp>
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
    allocator_type & __a,
    size_type __n) [inline], [static], [constexpr]
```

Allocate memory.

##### Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

**allocate()** [4/4]

```
template<typename _Tp>
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
    allocator_type & __a,
    size_type __n,
    const_void_pointer __hint) [inline], [static], [constexpr]
```

Allocate memory.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

**Returns**

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)`

**construct()** [1/2]

```
static constexpr void std::allocator_traits< allocator< _Tp > >::construct (
    allocator< _Tp > & __a,
    _Tp * __p,
    _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`

**Parameters**

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

**construct()** [2/2]

```
template<typename _Tp>
template<typename _Up, typename... _Args>
static constexpr void std::allocator_traits< allocator< _Tp > >::construct (
    allocator_type & __a,
    _Up * __p,
    _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Up`

**Parameters**

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                                             |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for an object of type <code>_Up</code> . |
| <code>__args</code> | Constructor arguments.                                                                    |

---

Calls `__a.construct(__p, std::forward<_Args>(__args)...) in C++11, C++14 and C++17. Changed in C++20 to call std::construct_at(__p, std::forward<_Args>(__args)...) instead. References std::_Construct(), and std::forward().`

**deallocate()** [1/2]

```
static constexpr void std::allocator_traits< allocator< _Tp > >::deallocate (
    allocator< _Tp > & __a,
    pointer __p,
    size_type __n) [inline], [static], [constexpr]
```

Deallocate memory.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__a</code> | An allocator.                                  |
| <code>__p</code> | Pointer to the memory to deallocate.           |
| <code>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

**deallocate()** [2/2]

```
template<typename _Tp>
static constexpr void std::allocator_traits< allocator< _Tp > >::deallocate (
    allocator_type & __a,
    pointer __p,
    size_type __n) [inline], [static], [constexpr]
```

Deallocate memory.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__a</code> | An allocator.                                  |
| <code>__p</code> | Pointer to the memory to deallocate.           |
| <code>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

**destroy()** [1/2]

```
static constexpr void std::allocator_traits< allocator< _Tp > >::destroy (
    allocator< _Tp > & __a,
    _Tp * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Tp`.

**Parameters**

|       |                                  |
|-------|----------------------------------|
| $\_a$ | An allocator.                    |
| $\_p$ | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~Tp()`

**destroy()** [2/2]

```
template<typename _Tp>
template<typename _Up>
static constexpr void std::allocator_traits< allocator< _Tp > >::destroy (
    allocator_type & __a,
    _Up * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Up`.

**Parameters**

|       |                                  |
|-------|----------------------------------|
| $\_a$ | An allocator.                    |
| $\_p$ | Pointer to the object to destroy |

Calls `__a.destroy(__p)`.

**max\_size()** [1/2]

```
static constexpr size_type std::allocator_traits< allocator< _Tp > >::max_size (
    const allocator< _Tp > & __a) [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

**Parameters**

|       |               |
|-------|---------------|
| $\_a$ | An allocator. |
|-------|---------------|

**Returns**

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

**max\_size()** [2/2]

```
template<typename _Tp>
static constexpr size_type std::allocator_traits< allocator< _Tp > >::max_size (
    const allocator_type & __a) [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

## Parameters

|                        |               |
|------------------------|---------------|
| <code>_↔<br/>_a</code> | An allocator. |
|------------------------|---------------|

## Returns

`__a.max_size()`

**select\_on\_container\_copy\_construction()** [1/2]

```
static constexpr allocator< _Tp > std::allocator_traits< allocator< _Tp > >::select_on_container↔
_copy_construction (
    const allocator< _Tp > & __rhs) [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

## Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

**select\_on\_container\_copy\_construction()** [2/2]

```
template<typename _Tp>
static constexpr allocator_type std::allocator_traits< allocator< _Tp > >::select_on_container↔
_copy_construction (
    const allocator_type & __rhs) [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

## Returns

`__rhs`

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

**5.190 `std::allocator_traits< allocator< void > >` Struct Reference**

```
#include <memory>
```



## Public Types

- typedef [allocator](#)< void > [allocator\\_type](#)
- using [allocator\\_type](#)
- using [const\\_pointer](#)
- using [const\\_pointer](#)
- using [const\\_void\\_pointer](#)
- using [const\\_void\\_pointer](#)
- using [difference\\_type](#)
- using [difference\\_type](#)
- using [is\\_always\\_equal](#)
- using [is\\_always\\_equal](#)
- using [pointer](#)
- using [pointer](#)
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_swap](#)
- using [propagate\\_on\\_container\\_swap](#)
- using **rebind\_alloc**
- template<typename \_Up>  
using **rebind\_alloc**
- using **rebind\_traits**
- template<typename \_Up>  
using **rebind\_traits**
- using [size\\_type](#)
- using [size\\_type](#)
- typedef [allocator](#)< void >::value\_type [value\\_type](#)
- using [value\\_type](#)
- using [void\\_pointer](#)
- using [void\\_pointer](#)

## Static Public Member Functions

- static constexpr [pointer](#) [allocate](#) ([allocator](#)< void > &\_\_a, [size\\_type](#) \_\_n)
- static constexpr [pointer](#) [allocate](#) ([allocator](#)< void > &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#) \_\_hint)
- static void \* [allocate](#) ([allocator\\_type](#) &, [size\\_type](#), const void \* = nullptr) = delete
- static constexpr void [construct](#) ([allocator](#)< void > &\_\_a, \_Tp \* \_\_p, \_Args &&... \_\_args) noexcept(\_S\_nothrow\_construct< \_Tp, \_Args... >())
- template<typename \_Up, typename... \_Args>  
static constexpr void [construct](#) ([allocator\\_type](#) &, \_Up \* \_\_p, \_Args &&... \_\_args) noexcept(\_\_is\_nothrow\_new\_constructible< \_Up, \_Args... >)
- static constexpr void [deallocate](#) ([allocator](#)< void > &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- static void [deallocate](#) ([allocator\\_type](#) &, void \*, [size\\_type](#)) = delete
- static constexpr void [destroy](#) ([allocator](#)< void > &\_\_a, \_Tp \* \_\_p) noexcept(\_S\_nothrow\_destroy< \_Tp >())
- template<typename \_Up>  
static constexpr void [destroy](#) ([allocator\\_type](#) &, \_Up \* \_\_p) noexcept(is\_nothrow\_destructible< \_Up >::value)
- static constexpr [size\\_type](#) [max\\_size](#) (const [allocator](#)< void > &\_\_a) noexcept
- static [size\\_type](#) [max\\_size](#) (const [allocator\\_type](#) &) = delete
- static constexpr [allocator](#)< void > [select\\_on\\_container\\_copy\\_construction](#) (const [allocator](#)< void > &\_\_rhs)
- static constexpr [allocator\\_type](#) [select\\_on\\_container\\_copy\\_construction](#) (const [allocator\\_type](#) &\_\_rhs)

### 5.190.1 Detailed Description

Explicit specialization for `std::allocator<void>`

Since

C++11

See also

`std::allocator_traits`

### 5.190.2 Member Typedef Documentation

#### **allocator\_type** [1/2]

```
typedef allocator< void > std::allocator_traits< allocator< void > >::allocator_type
```

The allocator type.

#### **allocator\_type** [2/2]

```
using std::allocator_traits< allocator< void > >::allocator_type
```

The allocator type.

#### **const\_pointer** [1/2]

```
using std::allocator_traits< allocator< void > >::const_pointer
```

The allocator's const pointer type.

`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

#### **const\_pointer** [2/2]

```
using std::allocator_traits< allocator< void > >::const_pointer
```

The allocator's const pointer type.

#### **const\_void\_pointer** [1/2]

```
using std::allocator_traits< allocator< void > >::const_void_pointer
```

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

#### **const\_void\_pointer** [2/2]

```
using std::allocator_traits< allocator< void > >::const_void_pointer
```

The allocator's const void pointer type.

#### **difference\_type** [1/2]

```
using std::allocator_traits< allocator< void > >::difference_type
```

The allocator's difference type.

`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

#### **difference\_type** [2/2]

```
using std::allocator_traits< allocator< void > >::difference_type
```

The allocator's difference type.

**is\_always\_equal** [1/2]

```
using std::allocator_traits< allocator< void > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

Alloc::is\_always\_equal if that type exists, otherwise is\_empty<Alloc>::type

**is\_always\_equal** [2/2]

```
using std::allocator_traits< allocator< void > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

**pointer** [1/2]

```
using std::allocator_traits< allocator< void > >::pointer
```

The allocator's pointer type.

Alloc::pointer if that type exists, otherwise value\_type\*

**pointer** [2/2]

```
using std::allocator_traits< allocator< void > >::pointer
```

The allocator's pointer type.

**propagate\_on\_container\_copy\_assignment** [1/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.

Alloc::propagate\_on\_container\_copy\_assignment if that type exists, otherwise false\_type

**propagate\_on\_container\_copy\_assignment** [2/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.

**propagate\_on\_container\_move\_assignment** [1/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.

Alloc::propagate\_on\_container\_move\_assignment if that type exists, otherwise false\_type

**propagate\_on\_container\_move\_assignment** [2/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.

**propagate\_on\_container\_swap** [1/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_swap
```

How the allocator is propagated on swap.

Alloc::propagate\_on\_container\_swap if that type exists, otherwise false\_type

**propagate\_on\_container\_swap** [2/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_swap
```

How the allocator is propagated on swap.

**size\_type** [1/2]

using `std::allocator_traits< allocator< void > >::size_type`

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

**size\_type** [2/2]

using `std::allocator_traits< allocator< void > >::size_type`

The allocator's size type.

**value\_type** [1/2]

typedef `allocator< void >::value_type` `std::allocator_traits< allocator< void > >::value_type`

The allocated type.

**value\_type** [2/2]

using `std::allocator_traits< allocator< void > >::value_type`

The allocated type.

**void\_pointer** [1/2]

using `std::allocator_traits< allocator< void > >::void_pointer`

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

**void\_pointer** [2/2]

using `std::allocator_traits< allocator< void > >::void_pointer`

The allocator's void pointer type.

**5.190.3 Member Function Documentation****allocate()** [1/3]

```
static constexpr pointer std::allocator_traits< allocator< void > >::allocate (
    allocator< void > & __a,
    size_type __n) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

**Parameters**

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

**allocate()** [2/3]

```
static constexpr pointer std::allocator_traits< allocator< void > >::allocate (
    allocator< void > & __a,
    size_type __n,
    const_void_pointer __hint) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

## Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

## Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

**allocate()** [3/3]

```
static void * std::allocator_traits< allocator< void > >::allocate (
    allocator_type & ,
    size_type ,
    const void * = nullptr) [static], [delete]
```

`allocate` is ill-formed for `allocator<void>`

**construct()** [1/2]

```
static constexpr void std::allocator_traits< allocator< void > >::construct (
    allocator< void > & __a,
    _Tp * __p,
    _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`

## Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args)...)`  if that expression is well-formed, otherwise uses placement-new to construct an object of type `_Tp` at location `__p` from the arguments `__args...`

**construct()** [2/2]

```
template<typename _Up, typename... _Args>
static constexpr void std::allocator_traits< allocator< void > >::construct (
    allocator_type & ,
    _Up * __p,
    _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Up`

## Parameters

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                                             |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for an object of type <code>_Up</code> . |
| <code>__args</code> | Constructor arguments.                                                                    |

Calls `__a.construct(__p, std::forward<_Args>(__args)...)`  in C++11, C++14 and C++17. Changed in C++20 to call `std::construct_at(__p, std::forward<_Args>(__args)...)`  instead. References `std::_Construct()`, and `std::forward()`.

**deallocate()** [1/2]

```
static constexpr void std::allocator_traits< allocator< void > >::deallocate (
    allocator< void > & __a,
    pointer __p,
    size_type __n) [inline], [static], [constexpr]
```

Deallocate memory.

**Parameters**

|                     |                                                |
|---------------------|------------------------------------------------|
| $\leftarrow$<br>__a | An allocator.                                  |
| $\leftarrow$<br>__p | Pointer to the memory to deallocate.           |
| $\leftarrow$<br>__n | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

**deallocate()** [2/2]

```
static void std::allocator_traits< allocator< void > >::deallocate (
    allocator_type & ,
    void * ,
    size_type ) [static], [delete]
```

deallocate is ill-formed for `allocator<void>`

**destroy()** [1/2]

```
static constexpr void std::allocator_traits< allocator< void > >::destroy (
    allocator< void > & __a,
    _Tp * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Tp`.

**Parameters**

|                     |                                  |
|---------------------|----------------------------------|
| $\leftarrow$<br>__a | An allocator.                    |
| $\leftarrow$<br>__p | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

**destroy()** [2/2]

```
template<typename _Up>
static constexpr void std::allocator_traits< allocator< void > >::destroy (
    allocator_type & ,
    _Up * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Up`

**Parameters**

|                     |               |
|---------------------|---------------|
| $\leftarrow$<br>__a | An allocator. |
|---------------------|---------------|

## Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__p</code> | Pointer to the object to destroy |
|------------------|----------------------------------|

Invokes the destructor for `*__p`.

References [std::\\_Destroy\(\)](#).

**max\_size()** [1/2]

```
static constexpr size_type std::allocator_traits< allocator< void > >::max_size (
    const allocator< void > & __a) [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

## Parameters

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

## Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

**max\_size()** [2/2]

```
static size_type std::allocator_traits< allocator< void > >::max_size (
    const allocator_type & ) [static], [delete]
```

`max_size` is ill-formed for `allocator<void>`

**select\_on\_container\_copy\_construction()** [1/2]

```
static constexpr allocator< void > std::allocator_traits< allocator< void > >::select_on_
container_copy_construction (
    const allocator< void > & __rhs) [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

## Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

**select\_on\_container\_copy\_construction()** [2/2]

```
static constexpr allocator_type std::allocator_traits< allocator< void > >::select_on_container_
_copy_construction (
    const allocator_type & __rhs) [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.



**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

**Returns**

`__rhs`

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

## 5.191 `std::allocator_traits< pmr::polymorphic_allocator< _Tp > >` Struct Template Reference

```
#include <memory_resource>
```

**Public Types**

- typedef [pmr::polymorphic\\_allocator< \\_Tp >](#) [allocator\\_type](#)
- using [allocator\\_type](#)
- using [const\\_pointer](#)
- using [const\\_pointer](#)
- using [const\\_void\\_pointer](#)
- using [const\\_void\\_pointer](#)
- using [difference\\_type](#)
- using [difference\\_type](#)
- using [is\\_always\\_equal](#)
- using [is\\_always\\_equal](#)
- using [pointer](#)
- using [pointer](#)
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_swap](#)
- using **rebind\_alloc**
- template<typename \_Up>  
  using **rebind\_alloc**
- using **rebind\_traits**
- template<typename \_Up>  
  using **rebind\_traits**
- using [size\\_type](#)
- using [size\\_type](#)
- typedef [pmr::polymorphic\\_allocator< \\_Tp >::value\\_type](#) [value\\_type](#)
- using [value\\_type](#)
- using [void\\_pointer](#)
- using [void\\_pointer](#)

**Static Public Member Functions**

- static [pointer allocate](#) ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n)
- static [pointer allocate](#) ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#))
- static constexpr [pointer allocate](#) ([pmr::polymorphic\\_allocator](#)< \_Tp > &\_\_a, [size\\_type](#) \_\_n)
- static constexpr [pointer allocate](#) ([pmr::polymorphic\\_allocator](#)< \_Tp > &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#) \_\_hint)
- template<typename \_Up, typename... \_Args>  
static void [construct](#) ([allocator\\_type](#) &\_\_a, \_Up \*\_\_p, \_Args &&... \_\_args)
- static constexpr void [construct](#) ([pmr::polymorphic\\_allocator](#)< \_Tp > &\_\_a, \_Tp \*\_\_p, \_Args &&... \_\_args) noexcept(\_S\_nothrow\_construct< \_Tp, \_Args... >())
- static void [deallocate](#) ([allocator\\_type](#) &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- static constexpr void [deallocate](#) ([pmr::polymorphic\\_allocator](#)< \_Tp > &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- template<typename \_Up>  
static constexpr void [destroy](#) ([allocator\\_type](#) &, \_Up \*\_\_p) noexcept(is\_nothrow\_destructible< \_Up >::value)
- static constexpr void [destroy](#) ([pmr::polymorphic\\_allocator](#)< \_Tp > &\_\_a, \_Tp \*\_\_p) noexcept(\_S\_nothrow\_←destroy< \_Tp >())
- static constexpr [size\\_type](#) [max\\_size](#) (const [allocator\\_type](#) &) noexcept
- static constexpr [size\\_type](#) [max\\_size](#) (const [pmr::polymorphic\\_allocator](#)< \_Tp > &\_\_a) noexcept
- static constexpr [pmr::polymorphic\\_allocator](#)< \_Tp > [select\\_on\\_container\\_copy\\_construction](#) (const [pmr::polymorphic\\_allocator](#)< \_Tp > &\_\_rhs)
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_swap](#)
- static [allocator\\_type](#) [select\\_on\\_container\\_copy\\_construction](#) (const [allocator\\_type](#) &) noexcept

**5.191.1 Detailed Description**

template<typename \_Tp>  
struct std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >

Partial specialization for std::pmr::polymorphic\_allocator

Since

C++17

**5.191.2 Member Typedef Documentation**

**allocator\_type** [1/2]

```
typedef pmr::polymorphic_allocator< _Tp > std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocator_type
```

The allocator type.

**allocator\_type** [2/2]

```
template<typename _Tp>  
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocator_type
```

The allocator type.

**const\_pointer** [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_pointer
```

The allocator's const pointer type.

Alloc::const\_pointer if that type exists, otherwise pointer\_traits<pointer>::rebind<const value\_type>

**const\_pointer** [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_pointer
The allocator's const pointer type.
```

**const\_void\_pointer** [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_void_pointer
The allocator's const void pointer type.
Alloc::const_void_pointer if that type exists, otherwise pointer_traits<pointer>::rebind<const
void>
```

**const\_void\_pointer** [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_void_pointer
The allocator's const void pointer type.
```

**difference\_type** [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::difference_type
The allocator's difference type.
Alloc::difference_type if that type exists, otherwise pointer_traits<pointer>::difference←
_type
```

**difference\_type** [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::difference_type
The allocator's difference type.
```

**is\_always\_equal** [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::is_always_equal
Whether all instances of the allocator type compare equal.
Alloc::is_always_equal if that type exists, otherwise is_empty<Alloc>::type
```

**is\_always\_equal** [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::is_always_equal
Whether all instances of the allocator type compare equal.
```

**pointer** [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::pointer
The allocator's pointer type.
Alloc::pointer if that type exists, otherwise value_type*
```

**pointer** [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::pointer
The allocator's pointer type.
```

**propagate\_on\_container\_copy\_assignment** [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_copy_↵
assignment
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

**propagate\_on\_container\_copy\_assignment** [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_copy_↵
assignment
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

**propagate\_on\_container\_move\_assignment** [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_move_↵
assignment
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

**propagate\_on\_container\_move\_assignment** [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_move_↵
assignment
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

**propagate\_on\_container\_swap** [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_swap
```

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

**propagate\_on\_container\_swap** [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_swap
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

**size\_type** [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::size_type
```

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

**size\_type** [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::size_type
```

The allocator's size type.

**value\_type** [1/2]

```
typedef pmr::polymorphic_allocator< _Tp >::value_type std::allocator_traits< pmr::polymorphic_allocator<
_Tp > >::value_type
```

The allocated type.

#### value\_type [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::value_type
```

The allocated type.

#### void\_pointer [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.  
Alloc::void\_pointer if that type exists, otherwise pointer\_traits<pointer>::rebind<void>

#### void\_pointer [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.

### 5.191.3 Member Function Documentation

#### allocate() [1/4]

```
template<typename _Tp>
static pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
    allocator_type & __a,
    size_type __n) [inline], [static], [nodiscard]
```

Allocate memory.

##### Parameters

|                                                                                            |                                              |
|--------------------------------------------------------------------------------------------|----------------------------------------------|
| <br>__a | An allocator.                                |
| <br>__n | The number of objects to allocate space for. |

Calls a.allocate(n).

#### allocate() [2/4]

```
template<typename _Tp>
static pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
    allocator_type & __a,
    size_type __n,
    const_void_pointer ) [inline], [static], [nodiscard]
```

Allocate memory.

##### Parameters

|                                                                                            |                                              |
|--------------------------------------------------------------------------------------------|----------------------------------------------|
| <br>__a | An allocator.                                |
| <br>__n | The number of objects to allocate space for. |

**Returns**

Memory of suitable size and alignment for *n* objects of type `value_type`.

The third parameter is ignored..

Returns `a.allocate(n)`.

**allocate()** [3/4]

```
static constexpr pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
    pmr::polymorphic_allocator< _Tp > & __a,
    size_type __n) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

**Parameters**

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

**allocate()** [4/4]

```
static constexpr pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
    pmr::polymorphic_allocator< _Tp > & __a,
    size_type __n,
    const_void_pointer __hint) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

**Returns**

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

**construct()** [1/2]

```
template<typename _Tp>
template<typename _Up, typename... _Args>
static void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::construct (
    allocator_type & __a,
    _Up * __p,
    _Args &&... __args) [inline], [static]
```

Construct an object of type `_Up`

**Parameters**

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

## Parameters

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <code>__p</code>    | Pointer to memory of suitable size and alignment for an object of type <code>_Up</code> . |
| <code>__args</code> | Constructor arguments.                                                                    |

Calls `__a.construct(__p, std::forward<_Args>(__args)...) in C++11, C++14 and C++17. Changed in C++20 to call std::construct_at(__p, std::forward<_Args>(__args)...) instead. References std::forward\(\).`

**construct()** [2/2]

```
static constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::construct (
    pmr::polymorphic_allocator< _Tp > & __a,
    _Tp * __p,
    _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`

## Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

**deallocate()** [1/2]

```
template<typename _Tp>
static void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::deallocate (
    allocator_type & __a,
    pointer __p,
    size_type __n) [inline], [static]
```

Deallocate memory.

## Parameters

|                          |                                                |
|--------------------------|------------------------------------------------|
| <code>__↵<br/>__a</code> | An allocator.                                  |
| <code>__↵<br/>__p</code> | Pointer to the memory to deallocate.           |
| <code>__↵<br/>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n).`

**deallocate()** [2/2]

```
static constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::deallocate (
    pmr::polymorphic_allocator< _Tp > & __a,
    pointer __p,
    size_type __n) [inline], [static], [constexpr]
```

Deallocate memory.

## Parameters

|                        |                                                |
|------------------------|------------------------------------------------|
| <code>_↵<br/>_a</code> | An allocator.                                  |
| <code>_↵<br/>_p</code> | Pointer to the memory to deallocate.           |
| <code>_↵<br/>_n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

**destroy()** [1/2]

```
template<typename _Tp>
template<typename _Up>
static constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::destroy (
    allocator_type & ,
    _Up * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Up`

## Parameters

|                        |                                  |
|------------------------|----------------------------------|
| <code>_↵<br/>_a</code> | An allocator.                    |
| <code>_↵<br/>_p</code> | Pointer to the object to destroy |

Calls `p->_Up()`.

**destroy()** [2/2]

```
static constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::destroy (
    pmr::polymorphic_allocator< _Tp > & __a,
    _Tp * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Tp`.

## Parameters

|                        |                                  |
|------------------------|----------------------------------|
| <code>_↵<br/>_a</code> | An allocator.                    |
| <code>_↵<br/>_p</code> | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

**max\_size()** [1/2]

```
template<typename _Tp>
static constexpr size_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::max_size
(
    const allocator_type & ) [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.



**Returns**

```
numeric_limits<size_t>::max() / sizeof(value_type)
```

**max\_size()** [2/2]

```
static constexpr size_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::max_size
(
    const pmr::polymorphic_allocator< _Tp > & __a) [inline], [static], [constexpr],
[noexcept]
```

The maximum supported allocation size.

**Parameters**

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

**Returns**

```
__a.max_size() or numeric_limits<size_type>::max()
```

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

**select\_on\_container\_copy\_construction()** [1/2]

```
template<typename _Tp>
static allocator_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::select_on_
container_copy_construction (
    const allocator_type & ) [inline], [static], [noexcept]
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

**select\_on\_container\_copy\_construction()** [2/2]

```
static constexpr pmr::polymorphic_allocator< _Tp > std::allocator_traits< pmr::polymorphic_allocator<
_Tp > >::select_on_container_copy_construction (
    const pmr::polymorphic_allocator< _Tp > & __rhs) [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

**Returns**

```
__rhs.select_on_container_copy_construction() or __rhs
```

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

The documentation for this struct was generated from the following file:

- [memory\\_resource.h](#)

**5.192 \_\_gnu\_cxx::limit\_condition::always\_adjustor Struct Reference**

```
#include <throw_allocator.h>
```

### 5.192.1 Detailed Description

Always enter the condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.193 `__gnu_cxx::random_condition::always_adjustor` Struct Reference

```
#include <throw_allocator.h>
```

### 5.193.1 Detailed Description

Always enter the condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.194 `__gnu_cxx::annotate_base` Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::annotate_base`:



### Public Member Functions

- void **check** (size\_t label)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)
- void **check\_constructed** (size\_t label)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)

### Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

## Friends

- [std::ostream](#) & **operator**<< ([std::ostream](#) &os, const [annotate\\_base](#) &\_\_b)

### 5.194.1 Detailed Description

Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size. The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.195 `std::experimental::fundamentals_v1::any` Class Reference

```
#include <any>
```

### Public Member Functions

- [any](#) () noexcept
- template<typename [\\_ValueType](#), typename [\\_Tp](#) = [\\_Decay](#)<[\\_ValueType](#)>, typename [\\_Mgr](#) = [\\_Manager](#)<[\\_Tp](#)>, typename [enable\\_if](#)<[is\\_constructible](#)<[\\_Tp](#), [\\_ValueType](#) && >::value, bool >::type = true>  
[any](#) ([\\_ValueType](#) &&\_\_value)
- template<typename [\\_ValueType](#), typename [\\_Tp](#) = [\\_Decay](#)<[\\_ValueType](#)>, typename [\\_Mgr](#) = [\\_Manager](#)<[\\_Tp](#)>, typename [enable\\_if](#)<![is\\_constructible](#)<[\\_Tp](#), [\\_ValueType](#) && >::value, bool >::type = false>  
[any](#) ([\\_ValueType](#) &&\_\_value)
- [any](#) ([any](#) &&\_\_other) noexcept
- [any](#) (const [any](#) &\_\_other)
- [~any](#) ()
- void [clear](#) () noexcept
- bool [empty](#) () const noexcept
- template<typename [\\_ValueType](#)>  
[enable\\_if](#) <![is\\_same](#)< [any](#), [decay\\_t](#)< [\\_ValueType](#) > >::value, [any](#) & > [operator=](#) ([\\_ValueType](#) &&\_\_rhs)
- [any](#) & [operator=](#) ([any](#) &&\_\_rhs) noexcept
- [any](#) & [operator=](#) (const [any](#) &\_\_rhs)
- void [swap](#) ([any](#) &\_\_rhs) noexcept
- const [type\\_info](#) & [type](#) () const noexcept

### Static Public Member Functions

- template<typename [\\_Tp](#)>  
static constexpr bool [\\_\\_is\\_valid\\_cast](#) ()

## Friends

- template<typename [\\_Tp](#)>  
[enable\\_if](#) <[is\\_object](#)< [\\_Tp](#) >::value, void \* > [\\_\\_any\\_caster](#) (const [any](#) \*\_\_any)

### 5.195.1 Detailed Description

A type-safe container of any type.

An `any` object's state is either empty or it stores a contained object of `CopyConstructible` type.

### 5.195.2 Constructor & Destructor Documentation

#### any() [1/5]

```
std::experimental::fundamentals_v1::any::any () [inline], [noexcept]
```

Default constructor, creates an empty object.

#### any() [2/5]

```
std::experimental::fundamentals_v1::any::any (
    const any & __other) [inline]
```

Copy constructor, copies the state of \_\_other.

#### any() [3/5]

```
std::experimental::fundamentals_v1::any::any (
    any && __other) [inline], [noexcept]
```

Move constructor, transfer the state from \_\_other.

#### Postcondition

\_\_other.empty() (this postcondition is a GNU extension)

#### any() [4/5]

```
template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if<is_constructible< _Tp, _ValueType && >::value, bool >::type = true>
```

```
std::experimental::fundamentals_v1::any::any (
    _ValueType && __value) [inline]
```

Construct with a copy of \_\_value as the contained object.

#### any() [5/5]

```
template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if<!is_constructible< _Tp, _ValueType && >::value, bool >::type = false>
```

```
std::experimental::fundamentals_v1::any::any (
    _ValueType && __value) [inline]
```

Construct with a copy of \_\_value as the contained object.

#### ~any()

```
std::experimental::fundamentals_v1::any::~any () [inline]
```

Destructor, calls clear()

### 5.195.3 Member Function Documentation

#### clear()

```
void std::experimental::fundamentals_v1::any::clear () [inline], [noexcept]
```

If not empty, destroy the contained object.

#### empty()

```
bool std::experimental::fundamentals_v1::any::empty () const [inline], [nodiscard], [noexcept]
```

Reports whether there is a contained object or not.

**operator=()** [1/3]

```
template<typename _ValueType>
enable_if_t<!is_same< any, decay_t< _ValueType > >::value, any & > std::experimental::fundamentals_v1::any::operator= (
    _ValueType && __rhs) [inline]
```

Store a copy of `__rhs` as the contained object.

**operator=()** [2/3]

```
any & std::experimental::fundamentals_v1::any::operator= (
    any && __rhs) [inline], [noexcept]
```

Move assignment operator.

**Postcondition**

`__rhs.empty()` (not guaranteed for other implementations)

**operator=()** [3/3]

```
any & std::experimental::fundamentals_v1::any::operator= (
    const any & __rhs) [inline]
```

Copy the state of another object.

**swap()**

```
void std::experimental::fundamentals_v1::any::swap (
    any & __rhs) [inline], [noexcept]
```

Exchange state with another object.

**type()**

```
const type_info & std::experimental::fundamentals_v1::any::type () const [inline], [noexcept]
```

The `typeid` of the contained object, or `typeid(void)` if empty.

The documentation for this class was generated from the following file:

- [experimental/any](#)

## 5.196 `std::array<_Tp, _Nm>` Struct Template Reference

```
#include <array>
```

**Public Types**

- typedef const value\_type \* **const\_iterator**
- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **iterator**
- typedef value\_type \* **pointer**
- typedef value\_type & **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

**Public Member Functions**

- constexpr reference **at** (size\_type \_\_n)
- constexpr const\_reference **at** (size\_type \_\_n) const
- constexpr const\_reference **back** () const noexcept
- constexpr reference **back** () noexcept
- constexpr const\_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- constexpr [const\\_reverse\\_iterator](#) **crend** () const noexcept
- constexpr const\_pointer **data** () const noexcept
- constexpr pointer **data** () noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr void **fill** (const value\_type &\_\_u)
- constexpr const\_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- constexpr size\_type **max\_size** () const noexcept
- constexpr const\_reference **operator[]** (size\_type \_\_n) const noexcept
- constexpr reference **operator[]** (size\_type \_\_n) noexcept
- constexpr [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- constexpr [reverse\\_iterator](#) **rbegin** () noexcept
- constexpr [const\\_reverse\\_iterator](#) **rend** () const noexcept
- constexpr [reverse\\_iterator](#) **rend** () noexcept
- constexpr size\_type **size** () const noexcept
- constexpr void **swap** ([array](#) &\_\_other) noexcept(\_\_array\_traits<\_Tp, \_Nm >::\_Is\_nothrow\_swappable::value)

**Public Attributes**

- `__array_traits<_Tp, _Nm >::_Type` **\_M\_elems**

**5.196.1 Detailed Description**

**template<typename \_Tp, std::size\_t \_Nm>**

**struct** `std::array<_Tp, _Nm >`

A standard container for storing a fixed size sequence of elements.

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#).

Sets support random access iterators.

**Template Parameters**

|           |                                                  |
|-----------|--------------------------------------------------|
| <i>Tp</i> | Type of element. Required to be a complete type. |
| <i>Nm</i> | Number of elements.                              |

The documentation for this struct was generated from the following file:

- [array](#)

## 5.197 \_\_gnu\_pbds::associative\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::associative\_tag:



### 5.197.1 Detailed Description

Basic associative-container.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.198 std::atomic<\_Tp> Struct Template Reference

```
#include <atomic>
```

### Public Types

- using **value\_type**

### Public Member Functions

- constexpr **atomic** (*\_Tp* \_\_i) noexcept
- **atomic** (const [atomic](#) &)=delete
- bool **compare\_exchange\_strong** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_strong** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_strong** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatile noexcept
- bool **compare\_exchange\_weak** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatile noexcept
- *\_Tp* **exchange** (*\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- *\_Tp* **exchange** (*\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- *\_Tp* **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- *\_Tp* **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_Tp** () const noexcept
- **operator \_Tp** () const volatile noexcept
- *\_Tp* **operator=** (*\_Tp* \_\_i) noexcept

- `_Tp operator= (_Tp __i)` volatile noexcept
- `atomic & operator= (const atomic &)` volatile =delete
- `atomic & operator= (const atomic &)=delete`
- void `store (_Tp __i, memory_order __m=memory_order_seq_cst)` noexcept
- void `store (_Tp __i, memory_order __m=memory_order_seq_cst)` volatile noexcept

### 5.198.1 Detailed Description

**template<typename \_Tp>**  
**struct std::atomic<\_Tp>**

Generic atomic type, primary class template.

#### Template Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>_Tp</code> | Type to be made atomic, must be trivially copyable. |
|------------------|-----------------------------------------------------|

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.199 std::atomic<\_Tp \* > Struct Template Reference

```
#include <atomic>
```

### Public Types

- typedef `__atomic_base<_Tp * > __base_type`
- typedef `_Tp * __pointer_type`
- using `difference_type`
- using `value_type`
- using `value_type`

### Public Member Functions

- constexpr `atomic (__pointer_type __p)` noexcept
- constexpr `atomic (_Tp __i)` noexcept
- `atomic (const atomic &)=delete`
- `atomic (const atomic &)=delete`
- bool `compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2)` noexcept
- bool `compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2)` volatile noexcept
- bool `compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst)` noexcept
- bool `compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst)` volatile noexcept
- bool `compare_exchange_strong (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst)` noexcept
- bool `compare_exchange_strong (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst)` volatile noexcept
- bool `compare_exchange_strong (_Tp &__e, _Tp __i, memory_order __s, memory_order __f)` noexcept
- bool `compare_exchange_strong (_Tp &__e, _Tp __i, memory_order __s, memory_order __f)` volatile noexcept
- bool `compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2)` noexcept



- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool compare_exchange_weak (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) noexcept`
- `bool compare_exchange_weak (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool compare_exchange_weak (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) noexcept`
- `bool compare_exchange_weak (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) volatile noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `_Tp exchange (_Tp __i, memory_order __m=memory_order_seq_cst) noexcept`
- `_Tp exchange (_Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool is_lock_free () const noexcept`
- `bool is_lock_free () const noexcept`
- `bool is_lock_free () const volatile noexcept`
- `bool is_lock_free () const volatile noexcept`
- `_Tp load (memory_order __m=memory_order_seq_cst) const noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const noexcept`
- `_Tp load (memory_order __m=memory_order_seq_cst) const volatile noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const volatile noexcept`
- `operator __pointer_type () const noexcept`
- `operator __pointer_type () const volatile noexcept`
- `operator _Tp () const noexcept`
- `operator _Tp () const volatile noexcept`
- `__pointer_type operator++ () noexcept`
- `__pointer_type operator++ () volatile noexcept`
- `__pointer_type operator++ (int) noexcept`
- `__pointer_type operator++ (int) volatile noexcept`
- `__pointer_type operator+= (ptrdiff_t __d) noexcept`
- `__pointer_type operator+= (ptrdiff_t __d) volatile noexcept`
- `__pointer_type operator-- () noexcept`
- `__pointer_type operator-- () volatile noexcept`
- `__pointer_type operator-- (int) noexcept`
- `__pointer_type operator-- (int) volatile noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) volatile noexcept`
- `__pointer_type operator= (__pointer_type __p) noexcept`
- `__pointer_type operator= (__pointer_type __p) volatile noexcept`
- `_Tp operator= (_Tp __i) noexcept`
- `_Tp operator= (_Tp __i) volatile noexcept`
- `atomic & operator= (const atomic &) volatile =delete`
- `atomic & operator= (const atomic &) volatile =delete`
- `atomic & operator= (const atomic &) =delete`
- `atomic & operator= (const atomic &) =delete`

- void **store** (\_\_pointer\_type \_\_p, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_pointer\_type \_\_p, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- void **store** (\_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

#### Public Attributes

- \_\_base\_type **\_M\_b**

#### 5.199.1 Detailed Description

**template<typename \_Tp>**  
**struct std::atomic<\_Tp \* >**

Partial specialization for pointer types.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.200 std::atomic\_flag Struct Reference

```
#include <atomic_base.h>
```

#### Public Member Functions

- constexpr **atomic\_flag** (bool \_\_i) noexcept
- **atomic\_flag** (const [atomic\\_flag](#) &)=delete
- void **clear** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **clear** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- [atomic\\_flag](#) & **operator=** (const [atomic\\_flag](#) &) volatile =delete
- [atomic\\_flag](#) & **operator=** (const [atomic\\_flag](#) &)=delete
- bool **test** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- bool **test** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- bool **test\_and\_set** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **test\_and\_set** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

#### 5.200.1 Detailed Description

[atomic\\_flag](#)

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 5.201 std::auto\_ptr<\_Tp> Class Template Reference

```
#include <memory>
```

#### Public Types

- typedef \_Tp [element\\_type](#)

## Public Member Functions

- `auto_ptr (auto_ptr &__a) throw ()`
- `template<typename _Tp1>  
auto_ptr (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr (auto_ptr_ref< element_type > __ref) throw ()`
- `auto_ptr (element_type *__p=0) throw ()`
- `~auto_ptr ()`
- `element_type * get () const throw ()`
- `template<typename _Tp1>  
operator auto_ptr< _Tp1 > () throw ()`
- `template<typename _Tp1>  
operator auto_ptr_ref< _Tp1 > () throw ()`
- `element_type & operator* () const throw ()`
- `element_type * operator-> () const throw ()`
- `auto_ptr & operator= (auto_ptr &__a) throw ()`
- `template<typename _Tp1>  
auto_ptr & operator= (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr & operator= (auto_ptr_ref< element_type > __ref) throw ()`
- `element_type * release () throw ()`
- `void reset (element_type *__p=0) throw ()`

### 5.201.1 Detailed Description

`template<typename _Tp>  
class std::auto_ptr< _Tp >`

A simple smart pointer providing strict ownership semantics.  
The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the CopyConstructible requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the libstdc++ testsuite.  
`_GLIBCXX_RESOLVE_LIB_DEFECTS`

1. `auto_ptr<>` conversion issues These resolutions have all been incorporated.

**Deprecated** Deprecated in C++11, no longer in the standard since C++17. Use `unique_ptr` instead.

### 5.201.2 Member Typedef Documentation

#### `element_type`

`template<typename _Tp>  
typedef _Tp std::auto_ptr< _Tp >::element_type`  
The pointed-to type.

## 5.201.3 Constructor &amp; Destructor Documentation

**auto\_ptr()** [1/4]

```
template<typename _Tp>
std::auto_ptr<_Tp>::auto_ptr (
    element_type * __p = 0) throw ( )    [inline], [explicit]
```

An auto\_ptr is usually constructed from a raw pointer.

**Parameters**

|                                                                                          |                               |
|------------------------------------------------------------------------------------------|-------------------------------|
| <br>__p | A pointer (defaults to NULL). |
|------------------------------------------------------------------------------------------|-------------------------------|

This object now *owns* the object pointed to by \_\_p.

Referenced by [auto\\_ptr\(\)](#), [auto\\_ptr\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

**auto\_ptr()** [2/4]

```
template<typename _Tp>
std::auto_ptr<_Tp>::auto_ptr (
    auto_ptr<_Tp> & __a) throw ( )    [inline]
```

An auto\_ptr can be constructed from another auto\_ptr.

**Parameters**

|                                                                                          |                                    |
|------------------------------------------------------------------------------------------|------------------------------------|
| <br>__a | Another auto_ptr of the same type. |
|------------------------------------------------------------------------------------------|------------------------------------|

This object now *owns* the object previously owned by \_\_a, which has given up ownership.

References [auto\\_ptr\(\)](#).

**auto\_ptr()** [3/4]

```
template<typename _Tp>
template<typename _Tp1>
std::auto_ptr<_Tp>::auto_ptr (
    auto_ptr<_Tp1> & __a) throw ( )    [inline]
```

An auto\_ptr can be constructed from another auto\_ptr.

**Parameters**

|                                                                                            |                                                   |
|--------------------------------------------------------------------------------------------|---------------------------------------------------|
| <br>__a | Another auto_ptr of a different but related type. |
|--------------------------------------------------------------------------------------------|---------------------------------------------------|

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by \_\_a, which has given up ownership.

References [auto\\_ptr\(\)](#).

**~auto\_ptr()**

```
template<typename _Tp>
std::auto_ptr<_Tp>::~~auto_ptr ( )    [inline]
```

When the auto\_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get ()` is NULL), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if `_Tp::~~_Tp()` throws, but this is prohibited. [17.4.3.6]/2

**auto\_ptr()** [4/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
    auto_ptr_ref< element_type > __ref) throw ( )    [inline]
```

Automatic conversions.

These operations are supposed to convert an auto\_ptr into and from an auto\_ptr\_ref automatically as needed. This would allow constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(...);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(...);
```

But it doesn't work, and won't be fixed. For further details see <https://cplusplus.github.io/←LWG/lwg-closed.html#463>

**5.201.4 Member Function Documentation****get()**

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::get ( ) const throw ( )    [inline]
```

Bypassing the smart pointer.

**Returns**

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

**Note**

This auto\_ptr still owns the memory.

**operator\*()**

```
template<typename _Tp>
element_type & std::auto_ptr< _Tp >::operator* ( ) const throw ( )    [inline]
```

Smart pointer dereferencing.

If this auto\_ptr no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

**operator->()**

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::operator-> ( ) const throw ( )    [inline]
```

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.

**operator=()** [1/2]

```
template<typename _Tp>
auto_ptr & std::auto_ptr< _Tp >::operator= (
    auto_ptr< _Tp > & __a) throw ( )    [inline]
```

auto\_ptr assignment operator.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__a</code> | Another <code>auto_ptr</code> of the same type. |
|------------------|-------------------------------------------------|

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

References [auto\\_ptr\(\)](#), and [reset\(\)](#).

**operator=()** [2/2]

```
template<typename _Tp>
template<typename _Tp1>
auto_ptr & std::auto_ptr<_Tp>::operator= (
    auto_ptr<_Tp1> & __a) throw ( )    [inline]
auto_ptr assignment operator.
```

## Parameters

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>__a</code> | Another <code>auto_ptr</code> of a different but related type. |
|------------------|----------------------------------------------------------------|

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

References [auto\\_ptr\(\)](#), and [reset\(\)](#).

**release()**

```
template<typename _Tp>
element_type * std::auto_ptr<_Tp>::release ( ) throw ( )    [inline]
Bypassing the smart pointer.
```

## Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

## Note

This `auto_ptr` no longer owns the memory. When this object goes out of scope, nothing will happen.

**reset()**

```
template<typename _Tp>
void std::auto_ptr<_Tp>::reset (
    element_type * __p = 0) throw ( )    [inline]
```

Forcibly deletes the managed object.

## Parameters

|                  |                               |
|------------------|-------------------------------|
| <code>__p</code> | A pointer (defaults to NULL). |
|------------------|-------------------------------|

This object now *owns* the object pointed to by `__p`. The previous object has been deleted.  
 Referenced by [operator=\(\)](#), and [operator=\(\)](#).

The documentation for this class was generated from the following files:

- [shared\\_ptr\\_base.h](#)
- [auto\\_ptr.h](#)

## 5.202 `std::auto_ptr_ref<_Tp1>` Struct Template Reference

```
#include <auto_ptr.h>
```

### Public Member Functions

- `auto_ptr_ref(_Tp1 *__p)`

### Public Attributes

- `_Tp1 * _M_ptr`

#### 5.202.1 Detailed Description

```
template<typename _Tp1>
struct std::auto_ptr_ref<_Tp1>
```

A wrapper class to provide `auto_ptr` with reference semantics. For example, an `auto_ptr` can be assigned (or constructed from) the result of a function which returns an `auto_ptr` by value.

All the `auto_ptr_ref` stuff should happen behind the scenes.

The documentation for this struct was generated from the following file:

- [auto\\_ptr.h](#)

## 5.203 `std::back_insert_iterator<_Container>` Class Template Reference

```
#include <stl_iterator.h>
```

Inheritance diagram for `std::back_insert_iterator<_Container>`:



## Public Types

- typedef `_Container` `container_type`
- using `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

## Public Member Functions

- constexpr `back_insert_iterator` (`_Container` &\_\_x)
- constexpr `back_insert_iterator` & `operator*` ()
- constexpr `back_insert_iterator` & `operator++` ()
- constexpr `back_insert_iterator` `operator++` (int)
- constexpr `back_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- constexpr `back_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_value)

## Protected Attributes

- `_Container` \* `container`

### 5.203.1 Detailed Description

`template<typename _Container>`  
**class** `std::back_insert_iterator<_Container>`

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using `push_back`.

Tip: Using the `back_inserter` function to create these iterators can save typing.

### 5.203.2 Member Typedef Documentation

#### `container_type`

```
template<typename _Container>
typedef _Container std::back_insert_iterator<_Container>::container_type
```

A nested typedef for the type of whatever container you used.

#### `iterator_category`

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator↵
_category [inherited]
```

One of the [tag types](#).

#### `pointer`

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-`value_type`.

#### `reference`

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-`value_type`.



**value\_type**

typedef void `std::iterator`< `output_iterator_tag`, void, void, void, void >::value\_type [inherited]  
 The type "pointed to" by the iterator.

**5.203.3 Constructor & Destructor Documentation****back\_insert\_iterator()**

```
template<typename _Container>
std::back_insert_iterator< _Container >::back_insert_iterator (
    _Container & __x) [inline], [explicit], [constexpr]
```

The only way to create this iterator is with a container.

References `std::__addressof()`.

**5.203.4 Member Function Documentation****operator\*()**

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator* () [inline], [nodiscard],
[constexpr]
```

Simply returns `*this`.

**operator++() [1/2]**

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator++ () [inline], [constexpr]
```

Simply returns `*this`. (This iterator does not *move*.)

**operator++() [2/2]**

```
template<typename _Container>
back_insert_iterator std::back_insert_iterator< _Container >::operator++ (
    int ) [inline], [constexpr]
```

Simply returns `*this`. (This iterator does not *move*.)

**operator=()**

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator= (
    const typename _Container::value_type & __value) [inline], [constexpr]
```

**Parameters**

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 5.204 std::bad\_alloc Class Reference

```
#include <new>
```

Inheritance diagram for std::bad\_alloc:



### Public Member Functions

- **bad\_alloc** (const [bad\\_alloc](#) &)=default
- **bad\_alloc** & **operator=** (const [bad\\_alloc](#) &)=default
- virtual const char \* **what** () const throw ()

#### 5.204.1 Detailed Description

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

#### 5.204.2 Member Function Documentation

##### **what()**

```
virtual const char * std::bad_alloc::what () const throw ( ) [virtual]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [new](#)

## 5.205 std::experimental::fundamentals\_v1::bad\_any\_cast Class Reference

```
#include <any>
```

Inheritance diagram for `std::experimental::fundamentals_v1::bad_any_cast`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 5.205.1 Detailed Description

Exception class thrown by a failed `any_cast`.

#### 5.205.2 Member Function Documentation

##### `what()`

```
virtual const char * std::experimental::fundamentals_v1::bad_any_cast::what () const [inline],  
[virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::bad\\_cast](#).

The documentation for this class was generated from the following file:

- [experimental/any](#)

### 5.206 `std::bad_cast` Class Reference

```
#include <typeinfo>
```

Inheritance diagram for `std::bad_cast`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 5.206.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

#### 5.206.2 Member Function Documentation

##### `what()`

```
virtual const char * std::bad_cast::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.207 `std::bad_exception` Class Reference

```
#include <exception>
```

Inheritance diagram for `std::bad_exception`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 5.207.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

#### 5.207.2 Member Function Documentation

##### **what()**

```
virtual const char * std::bad_exception::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [exception](#)

### 5.208 `std::bad_function_call` Class Reference

```
#include <std_function.h>
```

Inheritance diagram for `std::bad_function_call`:



**Public Member Functions**

- `const char * what () const noexcept`

**5.208.1 Detailed Description**

Exception class thrown when class template function's `operator()` is called with an empty target.

**5.208.2 Member Function Documentation****`what()`**

```
const char * std::bad_function_call::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

**5.209 std::experimental::fundamentals\_v1::bad\_optional\_access Class Reference**

```
#include <optional>
```

Inheritance diagram for `std::experimental::fundamentals_v1::bad_optional_access`:

**Public Member Functions**

- **`bad_optional_access`** (`const char * __arg`)
- `virtual const char * what () const noexcept`

**5.209.1 Detailed Description**

Exception class thrown when a disengaged optional object is dereferenced.

### 5.209.2 Member Function Documentation

#### what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [experimental/optional](#)

### 5.210 std::bad\_typeid Class Reference

```
#include <typeinfo>
```

Inheritance diagram for std::bad\_typeid:



#### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 5.210.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

#### 5.210.2 Member Function Documentation

##### what()

```
virtual const char * std::bad_typeid::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

### 5.211 std::bad\_weak\_ptr Class Reference

```
#include <shared_ptr_base.h>
```

Inheritance diagram for `std::bad_weak_ptr`:



### Public Member Functions

- virtual `char const * what () const` `noexcept`

#### 5.211.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

#### 5.211.2 Member Function Documentation

##### `what()`

```
virtual char const * std::bad_weak_ptr::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [shared\\_ptr\\_base.h](#)

### 5.212 \_\_gnu\_parallel::balanced\_quicksort\_tag Struct Reference

```
#include <tags.h>
```



Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



### Public Member Functions

- `balanced_quicksort_tag` (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

#### 5.212.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

#### 5.212.2 Member Function Documentation

##### `__get_num_threads()`

`_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () [inline], [inherited]

Find out desired number of threads.

##### Returns

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

##### `set_num_threads()`

void `__gnu_parallel::parallel_tag::set_num_threads` (  
     `_ThreadIndex` `__num_threads`) [inline], [inherited]

Set the desired number of threads.

##### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.213 `__gnu_parallel::balanced_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::balanced_tag`:



### Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

#### 5.213.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

#### 5.213.2 Member Function Documentation

##### `__get_num_threads()`

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads() [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

##### `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads(  
    \_ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

##### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.214 `std::tr2::bases<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- typedef `__reflection_typelist<__bases(_Tp)...>` **type**

#### 5.214.1 Detailed Description

```
template<typename _Tp>
struct std::tr2::bases<_Tp>
```

Sequence abstraction metafunctions for manipulating a typelist.

Enumerate all the base classes of a class. Form of a typelist.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 5.215 `__gnu_pbds::basic_branch<Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc>` Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for `__gnu_pbds::basic_branch<Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc>`:



### Public Types

- typedef `Node_Update` **node\_update**

### Protected Member Functions

- **basic\_branch** (const [basic\\_branch](#) &other)
- template<typename T0>  
**basic\_branch** (T0 t0)
- template<typename T0, typename T1>  
**basic\_branch** (T0 t0, T1 t1)
- template<typename T0, typename T1, typename T2>  
**basic\_branch** (T0 t0, T1 t1, T2 t2)
- template<typename T0, typename T1, typename T2, typename T3>  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0, typename T1, typename T2, typename T3, typename T4>  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)

- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5>`  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>`  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

### 5.215.1 Detailed Description

`template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename _Alloc>`

**class** `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >`

A branched, tree-like (tree, trie) container abstraction.

#### Template Parameters

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                           |
| <i>Mapped</i>      | Map type.                                                           |
| <i>Tag</i>         | Instantiating data structure type, see <code>container_tag</code> . |
| <i>Node_Update</i> | Updates nodes, restores invariants.                                 |
| <i>Policy_Tl</i>   | Policy typelist.                                                    |
| <i>_Alloc</i>      | Allocator type.                                                     |

Base is dispatched at compile time via `Tag`, from the following choices: `tree_tag`, `trie_tag`, and their descendants.

Base choices are: `detail::ov_tree_map`, `detail::rb_tree_map`, `detail::splay_tree_map`, and `detail::pat_trie_map`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.216 `__gnu_pbds::basic_branch_tag` Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::basic_branch_tag`:



### 5.216.1 Detailed Description

Basic branch structure.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.217 `std::basic_filebuf<_CharT, _Traits>` Class Template Reference

```
#include <fstream>
```

Inheritance diagram for `std::basic_filebuf<_CharT, _Traits>`:



## Public Types

- typedef `codecvt< char_type, char, __state_type > __codecvt_type`
- typedef `__basic_file< char > __file_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `traits_type::state_type __state_type`
- typedef `basic_streambuf< char_type, traits_type > __streambuf_type`
- typedef `_CharT char_type`
- typedef `traits_type::int_type int_type`
- typedef `traits_type::off_type off_type`
- typedef `traits_type::pos_type pos_type`
- typedef `_Traits traits_type`

## Public Member Functions

- `basic_filebuf()`
- `basic_filebuf(basic_filebuf &&)`
- `basic_filebuf(const basic_filebuf &)=delete`
- `virtual ~basic_filebuf()`
- `__filebuf_type * close()`
- `locale getloc() const`
- `streamsize in_avail()`
- `bool is_open() const throw()`
- `template<typename _Path>`  
`_If_fs_path< _Path, __filebuf_type * > open(const _Path &__s, ios_base::openmode __mode)`
- `__filebuf_type * open(const char * __s, ios_base::openmode __mode)`
- `__filebuf_type * open(const std::string & __s, ios_base::openmode __mode)`

- `basic_filebuf` & `operator=` (`basic_filebuf` &&)
- `basic_filebuf` & `operator=` (const `basic_filebuf` &)=delete
- `locale pubimbue` (const `locale` &\_\_loc)
- `int_type sbumpc` ()
- `int_type sgetc` ()
- `streamsize sgetn` (char\_type \*\_\_s, `streamsize` \_\_n)
- `int_type snextc` ()
- `int_type sputbackc` (char\_type \_\_c)
- `int_type sputc` (char\_type \_\_c)
- `streamsize sputn` (const char\_type \*\_\_s, `streamsize` \_\_n)
- `int_type sungetc` ()
- void `swap` (`basic_filebuf` &)
- `basic_streambuf` \* `pubsetbuf` (char\_type \*\_\_s, `streamsize` \_\_n)
- `pos_type pubseekoff` (off\_type \_\_off, `ios_base::seekdir` \_\_way, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
- `pos_type pubseekpos` (pos\_type \_\_sp, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
- `int pubsync` ()

### Protected Member Functions

- void `__safe_gbump` (`streamsize` \_\_n)
- void `__safe_pbump` (`streamsize` \_\_n)
- void `_M_allocate_internal_buffer` ()
- bool `_M_convert_to_external` (char\_type \*, `streamsize`)
- void `_M_create_pback` ()
- void `_M_destroy_internal_buffer` () throw ()
- void `_M_destroy_pback` () throw ()
- int `_M_get_ext_pos` (\_\_state\_type &\_\_state)
- `pos_type _M_seek` (off\_type \_\_off, `ios_base::seekdir` \_\_way, \_\_state\_type \_\_state)
- void `_M_set_buffer` (`streamsize` \_\_off)
- bool `_M_terminate_output` ()
- void `gbump` (int \_\_n)
- virtual void `imbue` (const `locale` &\_\_loc)
- virtual `int_type overflow` (int\_type \_\_c=\_Traits::eof())
- virtual `int_type pbackfail` (int\_type \_\_c=\_Traits::eof())
- void `pbump` (int \_\_n)
- virtual `pos_type seekoff` (off\_type \_\_off, `ios_base::seekdir` \_\_way, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
- virtual `pos_type seekpos` (pos\_type \_\_pos, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
- virtual `__streambuf_type * setbuf` (char\_type \*\_\_s, `streamsize` \_\_n)
- void `setg` (char\_type \*\_\_gbeg, char\_type \*\_\_gnext, char\_type \*\_\_gend)
- void `setp` (char\_type \*\_\_pbeg, char\_type \*\_\_pend)
- virtual `streamsize showmanyc` ()
- void `swap` (`basic_streambuf` &\_\_sb)
- virtual `int sync` ()
- virtual `int_type uflow` ()
- virtual `int_type underflow` ()
- virtual `streamsize xsgetn` (char\_type \*\_\_s, `streamsize` \_\_n)
- virtual `streamsize xspn` (const char\_type \*\_\_s, `streamsize` \_\_n)
- char\_type \* `eback` () const

- char\_type \* [gptr](#) () const
- char\_type \* [egptr](#) () const

- char\_type \* [pbase](#) () const
- char\_type \* [pptr](#) () const
- char\_type \* [epptr](#) () const

#### Protected Attributes

- char\_type \* [\\_M\\_buf](#)
  - bool [\\_M\\_buf\\_allocated](#)
  - [locale](#) [\\_M\\_buf\\_locale](#)
  - size\_t [\\_M\\_buf\\_size](#)
  - const [\\_\\_codecvt\\_type](#) \* [\\_M\\_codecvt](#)
  - char \* [\\_M\\_ext\\_buf](#)
  - [streamsize](#) [\\_M\\_ext\\_buf\\_size](#)
  - char \* [\\_M\\_ext\\_end](#)
  - const char \* [\\_M\\_ext\\_next](#)
  - [\\_\\_file\\_type](#) [\\_M\\_file](#)
  - char\_type \* [\\_M\\_in\\_beg](#)
  - char\_type \* [\\_M\\_in\\_cur](#)
  - char\_type \* [\\_M\\_in\\_end](#)
  - [\\_\\_c\\_lock](#) [\\_M\\_lock](#)
  - [ios\\_base::openmode](#) [\\_M\\_mode](#)
  - char\_type \* [\\_M\\_out\\_beg](#)
  - char\_type \* [\\_M\\_out\\_cur](#)
  - char\_type \* [\\_M\\_out\\_end](#)
  - bool [\\_M\\_reading](#)
  - [\\_\\_state\\_type](#) [\\_M\\_state\\_beg](#)
  - [\\_\\_state\\_type](#) [\\_M\\_state\\_cur](#)
  - [\\_\\_state\\_type](#) [\\_M\\_state\\_last](#)
  - bool [\\_M\\_writing](#)
- 
- char\_type [\\_M\\_pback](#)
  - char\_type \* [\\_M\\_pback\\_cur\\_save](#)
  - char\_type \* [\\_M\\_pback\\_end\\_save](#)
  - bool [\\_M\\_pback\\_init](#)

#### Friends

- class [ios\\_base](#)

#### 5.217.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_filebuf< _CharT, _Traits >
```

The actual work of input and output (for files).

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Requirements on `traits_type`, specific to this class:

- `traits_type::pos_type` must be `fpos<traits_type::state_type>`
- `traits_type::off_type` must be `streamoff`
- `traits_type::state_type` must be Assignable and DefaultConstructible,
- `traits_type::state_type()` must be the initial state for `codecvt`.

## 5.217.2 Constructor &amp; Destructor Documentation

**basic\_filebuf()**

```
template<typename _CharT, typename _Traits>
std::basic_filebuf< _CharT, _Traits >::basic_filebuf ()
```

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

References `_M_buf`, `std::basic_streambuf< _CharT, _Traits >::_M_buf_locale`, `_M_buf_size`, `_M_ext_buf`, `_M_ext_buf_size`, `_M_ext_next`, `_M_mode`, `_M_pback`, `_M_pback_cur_save`, `_M_pback_end_save`, `_M_pback_init`, and `_M_reading`.

Referenced by `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `close()`.

**~basic\_filebuf()**

```
template<typename _CharT, typename _Traits>
virtual std::basic_filebuf< _CharT, _Traits >::~~basic_filebuf () [inline], [virtual]
```

The destructor closes the file first.

## 5.217.3 Member Function Documentation

**\_M\_create\_pback()**

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_create_pback () [inline], [protected]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Referenced by `pbackfail()`.

**\_M\_destroy\_pback()**

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw () [inline], [protected]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Referenced by `overflow()`, `seekoff()`, `seekpos()`, `underflow()`, and `xsgetn()`.

**\_M\_set\_buffer()**

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (
    streamsize __off) [inline], [protected]
```



This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `egptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Referenced by `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, `close()`, `imbue()`, `open()`, `overflow()`, `pbackfail()`, `underflow()`, `xsgetn()`, and `xspn()`.

## close()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
()
```

Closes the currently associated file.

### Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

References `basic_filebuf()`, `_M_mode`, `_M_pback_init`, `_M_reading`, `_M_set_buffer()`, and `is_open()`.

Referenced by `open()`.

## eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

## egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

**epptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic\_streambuf<\_CharT, \_Traits>::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#), and [xsputn\(\)](#).

**gbump()**

```
template<typename _CharT, typename _Traits>
void std::basic\_streambuf<\_CharT, \_Traits>::gbump (
    int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::pbackfail\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#).

**getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic\_streambuf<\_CharT, \_Traits>::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

**gptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic\_streambuf<\_CharT, \_Traits>::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::imbue\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::showmanyc\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::underflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::underflow\(\)](#), [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::underflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#), and [xsgetn\(\)](#).

### imbue()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::imbue (
    const locale & __loc) [protected], [virtual]
```

Changes translations.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

References [\\_M\\_ext\\_buf](#), [\\_M\\_ext\\_next](#), [\\_M\\_mode](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), [std::ios\\_base::cur](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::gptr\(\)](#), [is\\_open\(\)](#), and [seekoff\(\)](#).

### in\_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

### is\_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf<_CharT, _Traits>::is_open () const throw () [inline], [nodiscard]
```

Returns true if the external file is open.

Referenced by [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [close\(\)](#), [imbue\(\)](#), [open\(\)](#), [seekoff\(\)](#), [seekpos\(\)](#), [setbuf\(\)](#), and [showmanyc\(\)](#).

### open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path<_Path, __filebuf_type * > std::basic_filebuf<_CharT, _Traits>::open (
    const _Path & __s,
    ios_base::openmode __mode) [inline]
```

Opens an external file.

## Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The name of the file, as a filesystem::path. |
| <code>__mode</code> | The open mode flags.                         |

## Returns

`this` on success, NULL on failure

**open()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::__filebuf_type * std::basic_filebuf<_CharT, _Traits>::open (
    const char * __s,
    ios_base::openmode __mode)
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

## Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

| ios_base Flag combination |    |     |       |     | stdio equivalent |
|---------------------------|----|-----|-------|-----|------------------|
| binary                    | in | out | trunc | app |                  |
|                           |    | +   |       |     | w                |
|                           |    | +   |       | +   | a                |
|                           |    |     |       | +   | a                |
|                           |    | +   | +     |     | w                |
|                           | +  |     |       |     | r                |
|                           | +  | +   |       |     | r+               |
|                           | +  | +   | +     |     | w+               |
|                           | +  | +   |       | +   | a+               |
|                           | +  |     |       | +   | a+               |
| +                         |    | +   |       |     | wb               |
| +                         |    | +   |       | +   | ab               |
| +                         |    |     |       | +   | ab               |
| +                         |    | +   | +     |     | wb               |
| +                         | +  |     |       |     | rb               |
| +                         | +  | +   |       |     | r+b              |
| +                         | +  | +   | +     |     | w+b              |
| +                         | +  | +   |       | +   | a+b              |
| +                         | +  |     |       | +   | a+b              |

References `_M_mode`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::ate`, `close()`, `std::ios_base::end`, `is_open()`, and `seekoff()`.

**open()** [3/3]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
    const std::string & __s,
    ios_base::openmode __mode) [inline]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

**Returns**

`this` on success, NULL on failure

**overflow()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (
    int_type __c = _Traits::eof()) [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_buf_size`, `_M_destroy_pback()`, `_M_mode`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::app`, `std::ios_base::cur`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pbuf()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

Referenced by `pbackfail()`, `sync()`, `underflow()`, and `xsgetn()`.

**pbackfail()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (
    int_type __c = _Traits::eof()) [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_↔</code> | The character to be inserted back into the sequence. |
| <code>_C</code> |                                                      |

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

References [\\_M\\_create\\_pback\(\)](#), [\\_M\\_mode](#), [\\_M\\_pback\\_init](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), [std::ios\\_base::cur](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::eback\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::gbump\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::in](#), [std::ios\\_base::in](#), [overflow\(\)](#), [seekoff\(\)](#), and [underflow\(\)](#).

**pbase()**

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic\_streambuf<\_CharT, \_Traits>::pbase\(\) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::sync\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

**pbump()**

```
template<typename _CharT, typename _Traits>
```

```
void std::basic\_streambuf<\_CharT, \_Traits>::pbump (  
    int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↔</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#).

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::xsputn()`, and `xsputn()`.

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived `imbue(__loc)`.

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekoff` function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

**pubsetbuf()**

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf<_CharT, _Traits>::pubsetbuf (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

**pubsync()**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf<_CharT, _Traits>::pubsync () [inline], [inherited]
```

Calls virtual `sync` function.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

**sbumpc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sbumpc () [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`.

**seekoff()**

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::pos_type std::basic_filebuf<_CharT, _Traits>::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

## Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_destroy_pback()`, `_M_reading`, `std::ios_base::cur`, `is_open()`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

Referenced by `imbue()`, `open()`, and `pbackfail()`.



**seekpos()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_destroy_pback()`, `std::ios_base::beg`, and `is_open()`.

**setbuf()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >↵
::setbuf (
    char_type * __s,
    streamsize __n) [protected], [virtual]
```

Manipulates the buffer.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <code>↵<br/>__s</code> | Pointer to a buffer area.  |
| <code>↵<br/>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.↵html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_buf`, `_M_buf_size`, and `is_open()`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setp (
    char_type * __pbeg,
    char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`.

**sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::ignore()`.

**sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

### showmanyc()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc () [protected], [virtual]
```

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::ios_base::binary`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::ios_base::in`, and `is_open()`.

### snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline], [inherited]
```

Getting the next character.

#### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< char_type, traits_type >::ignore()`, `std::basic_istream< char_type, traits_type >::operator>>()`, and `std::basic_istream< char_type, traits_type >::putback()`.

### sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
    char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

#### Returns

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
    char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
    const char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xsgputn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream< char_type, traits_type >::sentry::sentry()`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_filebuf< _CharT, _Traits >::sync () [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `overflow()`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

**uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Referenced by `xsgetn()`.

**underflow()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_buf_size`, `_M_destroy_pback()`, `_M_mode`, `_M_set_buffer()`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::ios_base::in`, and `overflow()`.

Referenced by `pbackfail()`.

**xsggetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsggetn (
    char_type * __s,
    streamsize __n) [protected], [virtual]
```

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_buf_size`, `_M_destroy_pback()`, `_M_mode`, `_M_pback_init`, `_M_reading`, `_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gbump()`, `std::basic_streambuf<_CharT, _Traits>::in`, `std::ios_base::in`, `overflow()`, `std::basic_streambuf<_CharT, _Traits>::setg()`, and `std::basic_streambuf<char_type, traits_type>::xsggetn()`.

**xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsputn (
    const char_type * __s,
    streamsize __n) [protected], [virtual]
```

Multiple character insertion.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_buf_size`, `_M_mode`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::app`, `std::basic_streambuf<_CharT, _Traits>::eptr()`, `std::ios_base::out`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, `std::basic_streambuf<_CharT, _Traits>::pptr()`, and `std::basic_streambuf<char_type, traits_type>::xsputn()`.

#### 5.217.4 Member Data Documentation

##### **\_M\_buf**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_filebuf< \_CharT, \_Traits >::\_M\_buf [protected]
```

Pointer to the beginning of internal buffer.

Referenced by [basic\\_filebuf\(\)](#), and [setbuf\(\)](#).

##### **\_M\_buf\_locale**

```
template<typename _CharT, typename _Traits>
locale std::basic\_streambuf< \_CharT, \_Traits >::\_M\_buf\_locale [protected], [inherited]
```

Current locale setting.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::basic\\_filebuf\(\)](#).

##### **\_M\_buf\_size**

```
template<typename _CharT, typename _Traits>
size_t std::basic\_filebuf< \_CharT, \_Traits >::\_M\_buf\_size [protected]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [setbuf\(\)](#), [underflow\(\)](#), [xsgetn\(\)](#), and [xspn\(\)](#).

##### **\_M\_ext\_buf**

```
template<typename _CharT, typename _Traits>
char* std::basic\_filebuf< \_CharT, \_Traits >::\_M\_ext\_buf [protected]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Referenced by [basic\\_filebuf\(\)](#), and [imbue\(\)](#).

##### **\_M\_ext\_buf\_size**

```
template<typename _CharT, typename _Traits>
streamsize std::basic\_filebuf< \_CharT, \_Traits >::\_M\_ext\_buf\_size [protected]
```

Size of buffer held by `_M_ext_buf`.

Referenced by [basic\\_filebuf\(\)](#).

##### **\_M\_ext\_next**

```
template<typename _CharT, typename _Traits>
const char* std::basic\_filebuf< \_CharT, \_Traits >::\_M\_ext\_next [protected]
```

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Referenced by [basic\\_filebuf\(\)](#), and [imbue\(\)](#).

##### **\_M\_in\_beg**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< \_CharT, \_Traits >::\_M\_in\_beg [protected], [inherited]
```

Start of get area.

**`_M_in_cur`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
Current read area.
```

**`_M_in_end`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
End of get area.
```

**`_M_mode`**

```
template<typename _CharT, typename _Traits>
ios\_base::openmode std::basic\_filebuf< _CharT, _Traits >::_M_mode [protected]
```

Place to stash in || out || in | out settings for current filebuf.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [showmanyc\(\)](#), [underflow\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

**`_M_out_beg`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
Start of put area.
```

**`_M_out_cur`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
Current put area.
```

**`_M_out_end`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
End of put area.
```

**`_M_pback`**

```
template<typename _CharT, typename _Traits>
char_type std::basic\_filebuf< _CharT, _Traits >::_M_pback [protected]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

**`_M_pback_cur_save`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_filebuf< _CharT, _Traits >::_M_pback_cur_save [protected]
```

Necessary bits for putback buffer management.



**Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

**\_M\_pback\_end\_save**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_filebuf< _CharT, _Traits >::_M_pback_end_save [protected]
Necessary bits for putback buffer management.
```

**Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

**\_M\_pback\_init**

```
template<typename _CharT, typename _Traits>
bool std::basic\_filebuf< _CharT, _Traits >::_M_pback_init [protected]
Necessary bits for putback buffer management.
```

**Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#), [close\(\)](#), [pbackfail\(\)](#), and [xsgetn\(\)](#).

**\_M\_reading**

```
template<typename _CharT, typename _Traits>
bool std::basic\_filebuf< _CharT, _Traits >::_M_reading [protected]
_M_reading == false && _M_writing == false for uncommitted mode; _M_reading == true for read mode; _M_writing
== true for write mode;
```

NB: \_M\_reading == true && \_M\_writing == true is unused.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [seekoff\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

The documentation for this class was generated from the following files:

- [fstream](#)
- [fstream.tcc](#)

**5.218 std::basic\_fstream< \_CharT, \_Traits > Class Template Reference**

```
#include <fstream>
```

Inheritance diagram for std::basic\_fstream< \_CharT, \_Traits >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< char_type, traits_type > __ios_type`
- typedef `basic_iostream< char_type, traits_type > __iostream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event __e, ios_base & __b, int __i)`
- typedef `traits_type::int_type int_type`

- typedef `_ios_istate` `istate`
- typedef `traits_type::off_type` `off_type`
- typedef `_ios_Openmode` `openmode`
- typedef `traits_type::pos_type` `pos_type`
- typedef `_ios_Seekdir` `seekdir`
- typedef `_Traits` `traits_type`

## Public Member Functions

- `basic_fstream` ()
- `basic_fstream` (`basic_fstream` &&\_\_rhs)
- template<typename `_Path`, typename `_Require` = `_If_fs_path<_Path>>`  
`basic_fstream` (const `_Path` &\_\_s, `ios_base::openmode` \_\_mode=`ios_base::in|ios_base::out`)
- `basic_fstream` (const `basic_fstream` &)=delete
- `basic_fstream` (const char \*\_\_s, `ios_base::openmode` \_\_mode=`ios_base::in|ios_base::out`)
- `basic_fstream` (const `std::string` &\_\_s, `ios_base::openmode` \_\_mode=`ios_base::in|ios_base::out`)
- `~basic_fstream` ()
- template<typename `_ValueT`>  
`basic_istream`< `_CharT`, `_Traits` > & `_M_extract` (`_ValueT` &\_\_v)
- const `locale` & `_M_getloc` () const
- template<typename `_ValueT`>  
`basic_ostream`< `_CharT`, `_Traits` > & `_M_insert` (`_ValueT` \_\_v)
- void `_M_setstate` (`istate` \_\_state)
- bool `bad` () const
- void `clear` (`istate` \_\_state=`goodbit`)
- void `close` ()
- `basic_ios` & `copyfmt` (const `basic_ios` &\_\_rhs)
- bool `eof` () const
- `istate exceptions` () const
- void `exceptions` (`istate` \_\_except)
- bool `fail` () const
- char\_type `fill` () const
- char\_type `fill` (char\_type \_\_ch)
- `fmtflags flags` () const
- `fmtflags flags` (`fmtflags` \_\_fmtfl)
- `__ostream_type` & `flush` ()
- `streamsize gcount` () const
- `basic_istream`< char > & `getline` (char\_type \*\_\_s, `streamsize` \_\_n, char\_type \_\_delim)
- `basic_istream`< wchar\_t > & `getline` (char\_type \*\_\_s, `streamsize` \_\_n, char\_type \_\_delim)
- `locale getloc` () const
- bool `good` () const
- `basic_istream`< char > & `ignore` (`streamsize` \_\_n)
- `basic_istream`< wchar\_t > & `ignore` (`streamsize` \_\_n)
- `basic_istream`< char > & `ignore` (`streamsize` \_\_n, int\_type \_\_delim)
- `basic_istream`< wchar\_t > & `ignore` (`streamsize` \_\_n, int\_type \_\_delim)
- `locale imbue` (const `locale` &\_\_loc)
- bool `is_open` ()
- bool `is_open` () const
- long & `isword` (int \_\_ix)
- char `narrow` (char\_type \_\_c, char \_\_dfault) const

- template<typename \_Path>  
\_If\_fs\_path< \_Path, void > open (const \_Path &\_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- void open (const char \*\_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- void open (const std::string &\_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- \_\_ostream\_type & operator<< (\_\_streambuf\_type \*\_\_sb)
- \_\_ostream\_type & operator<< (const void \*\_\_p)
- \_\_ostream\_type & operator<< (nullptr\_t)
- basic\_fstream & operator= (basic\_fstream &&\_\_rhs)
- basic\_fstream & operator= (const basic\_fstream &)=delete
- \_\_istream\_type & operator>> (\_\_streambuf\_type \*\_\_sb)
- \_\_istream\_type & operator>> (void \*&\_\_p)
- streamsize precision () const
- streamsize precision (streamsize \_\_prec)
- void \*& pword (int \_\_ix)
- \_\_filebuf\_type \* rdbuf () const
- basic\_streambuf< \_CharT, \_Traits > \* rdbuf (basic\_streambuf< \_CharT, \_Traits > \*\_\_sb)
- iostate rdstate () const
- void register\_callback (event\_callback \_\_fn, int \_\_index)
- \_\_ostream\_type & seekp (off\_type, ios\_base::seekdir)
- \_\_ostream\_type & seekp (pos\_type)
- fmtflags setf (fmtflags \_\_fmtfl)
- fmtflags setf (fmtflags \_\_fmtfl, fmtflags \_\_mask)
- void setstate (iostate \_\_state)
- void swap (basic\_fstream &\_\_rhs)
- pos\_type tellp ()
- basic\_ostream< \_CharT, \_Traits > \* tie () const
- basic\_ostream< \_CharT, \_Traits > \* tie (basic\_ostream< \_CharT, \_Traits > \*\_\_tiestr)
- void unsetf (fmtflags \_\_mask)
- char\_type widen (char \_\_c) const
- streamsize width () const
- streamsize width (streamsize \_\_wide)
  
- \_\_istream\_type & operator>> (\_\_istream\_type &)(\_\_pf)(\_\_istream\_type &))
- \_\_istream\_type & operator>> (\_\_ios\_type &)(\_\_pf)(\_\_ios\_type &))
- \_\_istream\_type & operator>> (ios\_base &)(\_\_pf)(ios\_base &))

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- \_\_istream\_type & operator>> (bool &\_\_n)
- \_\_istream\_type & operator>> (short &\_\_n)
- \_\_istream\_type & operator>> (unsigned short &\_\_n)
- \_\_istream\_type & operator>> (int &\_\_n)
- \_\_istream\_type & operator>> (unsigned int &\_\_n)
- \_\_istream\_type & operator>> (long &\_\_n)
- \_\_istream\_type & operator>> (unsigned long &\_\_n)

- `__istream_type & operator>>` (long long &\_\_n)
- `__istream_type & operator>>` (unsigned long long &\_\_n)
- `__istream_type & operator>>` (float &\_\_f)
- `__istream_type & operator>>` (double &\_\_f)
- `__istream_type & operator>>` (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get()`
- `__istream_type & get` (char\_type &\_\_c)
- `__istream_type & get` (char\_type \*\_\_s, streamsize \_\_n, char\_type \_\_delim)
- `__istream_type & get` (char\_type \*\_\_s, streamsize \_\_n)
- `__istream_type & get` (\_\_streambuf\_type &\_\_sb, char\_type \_\_delim)
- `__istream_type & get` (\_\_streambuf\_type &\_\_sb)
- `__istream_type & getline` (char\_type \*\_\_s, streamsize \_\_n, char\_type \_\_delim)
- `__istream_type & getline` (char\_type \*\_\_s, streamsize \_\_n)
- `__istream_type & ignore` (streamsize \_\_n, int\_type \_\_delim)
- `__istream_type & ignore` (streamsize \_\_n)
- `__istream_type & ignore` ()
- `int_type peek()`
- `__istream_type & read` (char\_type \*\_\_s, streamsize \_\_n)
- `streamsize readsome` (char\_type \*\_\_s, streamsize \_\_n)
- `__istream_type & putback` (char\_type \_\_c)
- `__istream_type & unget()`
- `int sync()`
- `pos_type tellg()`
- `__istream_type & seekg` (pos\_type)
- `__istream_type & seekg` (off\_type, ios\_base::seekdir)

- `operator bool` () const
- `bool operator!` () const

- `__ostream_type & operator<<` (\_\_ostream\_type &(\_\_pf)(\_\_ostream\_type &))
- `__ostream_type & operator<<` (\_\_ios\_type &(\_\_pf)(\_\_ios\_type &))
- `__ostream_type & operator<<` (ios\_base &(\_\_pf)(ios\_base &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`

- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Public Attributes

- class `__attribute__((__abi_tag__("cxx11")))` failure typedef `_ios_Fmtflags` `fmtflags`

### Static Public Attributes

- static const `openmode` `__noreplace`
- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`

- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) &\_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- template<typename [\\_ValueT](#)>  
[\\_\\_istream\\_type](#) & [\\_M\\_extract](#) ([\\_ValueT](#) &\_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- template<typename [\\_ValueT](#)>  
[\\_\\_ostream\\_type](#) & [\\_M\\_insert](#) ([\\_ValueT](#) \_\_v)
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) &\_\_rhs) noexcept
- void [init](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- void [move](#) ([basic\\_ios](#) &&\_\_rhs)
- void [move](#) ([basic\\_ios](#) &\_\_rhs)
- void [set\\_rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- void [swap](#) ([basic\\_ios](#) &\_\_rhs) noexcept
- void [swap](#) ([basic\\_istream](#) &\_\_rhs)
- void [swap](#) ([basic\\_istream](#) &\_\_rhs)
- void [swap](#) ([basic\\_ostream](#) &\_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const [\\_\\_ctype\\_type](#) \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- [char\\_type](#) [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [streamsize](#) [\\_M\\_gcount](#)
- [locale](#) [\\_M\\_ios\\_locale](#)

- `_Words` `_M_local_word` [`_S_local_word_size`]
- `const` `__num_get_type` \* `_M_num_get`
- `const` `__num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > \* `_M_streambuf`
- `iosstate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > \* `_M_tie`
- `streamsize` `_M_width`
- `_Words` \* `_M_word`
- `int` `_M_word_size`
- `_Words` `_M_word_zero`

### 5.218.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_fstream< _CharT, _Traits >
```

Controlling input and output for files.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from and writing to named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

### 5.218.2 Member Typedef Documentation

#### event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
The type of an event callback function.
```

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

#### iosstate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`



- eofbit
- failbit
- goodbit

### openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

### seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

## 5.218.3 Member Enumeration Documentation

### event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

## 5.218.4 Constructor & Destructor Documentation

### basic\_fstream() [1/4]

```
template<typename _CharT, typename _Traits>  
std::basic_fstream< _CharT, _Traits >::basic_fstream () [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

**basic\_fstream()** [2/4]

```
template<typename _CharT, typename _Traits>
std::basic_fstream<_CharT, _Traits>::basic_fstream (
    const char * __s,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Create an input/output file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.  |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

**basic\_fstream()** [3/4]

```
template<typename _CharT, typename _Traits>
std::basic_fstream<_CharT, _Traits>::basic_fstream (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Create an input/output file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.  |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

**basic\_fstream()** [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_fstream<_CharT, _Traits>::basic_fstream (
    const _Path & __s,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Create an input/output file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | filesystem::path specifying the filename.        |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

**~basic\_fstream()**

```
template<typename _CharT, typename _Traits>
std::basic_fstream<_CharT, _Traits>::~~basic_fstream () [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

## 5.218.5 Member Function Documentation

### **`_M_getloc()`**

const [locale](#) & std::ios\_base::\_M\_getloc () const [inline], [inherited]

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_date\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_weekday\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_year\(\)](#), [std::num\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), [std::time\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::get\(\)](#), and [std::time\\_put<\\_CharT, \\_Outlter>::put\(\)](#).

### **`bad()`**

template<typename \_CharT, typename \_Traits>

bool [std::basic\\_ios<\\_CharT, \\_Traits>::bad](#) () const [inline], [nodiscard], [inherited]

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

### **`clear()`**

template<typename \_CharT, typename \_Traits>

void [std::basic\\_ios<\\_CharT, \\_Traits>::clear](#) (  
    [iostate](#) \_\_state = [goodbit](#)) [inherited]

[Re]sets the error state.

#### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::exceptions\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::unget\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

### **`close()`**

template<typename \_CharT, typename \_Traits>

void [std::basic\\_fstream<\\_CharT, \\_Traits>::close](#) () [inline]

Close the file.

Calls [std::basic\\_filebuf::close\(\)](#). If that function fails, [failbit](#) is set in the stream's error state.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of \_\_rhs into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
```

```
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
Fast error checking.
```

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#), and [std::regex\\_traits< \\_Ch\\_type >::value\(\)](#).

**fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
Retrieves the empty character.
```

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by [copyfmt\(\)](#), [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#), and [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#).

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

## Returns

The format control flags for both input and output.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_put(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::subtract\_with\_carry\_engine< uint\_fast32\_t >, std::chrono::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), and std::subtract\_with\_carry\_engine< unsigned int >

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline], [inherited]
```

## Setting new format flags all at once.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

## Returns

The previous format control flags.

This function overwrites all the format flags with `fmtfl`.

## References [fmtflags](#).

## flush()

```
template<typename _CharT, typename _Traits>
```

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush () [inherited]
```

Synchronizing the stream buffer.

## Returns

```
*this
```

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits](#)

## gcount()

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream<_CharT, _Traits >::qcount () const [inline], [inherited]
```

Character counting.

## Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb, widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References `_M_gcount`, and `std::ios_base::goodbit`.



**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c) [inherited]
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s,__n,widen("\n"))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Simple multiple-character extraction.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

## Returns

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

## Note

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

**getline()** [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::getline (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

String extraction.

## Parameters

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

## Returns

`*this`

Returns `getline(__s,__n,widen("\n"))`.

**getline()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References [\\_M\\_gcount](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sgetc\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::snextc\(\)](#).

**getline()** [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale::global()`, the global C++ locale.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::money\\_put<\\_CharT, \\_Outiter>::do\\_put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::do\\_get\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

**ignore()** [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**ignore()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_streambuf<_CharT, _Traits>::rdbuf()`.

**ignore()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim) [inherited]
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

`*this`

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

References `_M_gcount`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_streambuf<_CharT, _Traits>::rdbuf()`.

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References `std::ios_base::getloc()`, `std::ios_base::imbue()`, and `rdbuf()`.

Referenced by `std::chrono::operator<<()`.

**init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_ios< char_type, traits_type >::basic_ios()`, and `std::basic_ostream< char_type, traits_type >::basic_ostream()`.

**is\_open()**

```
template<typename _CharT, typename _Traits>
bool std::basic_fstream< _CharT, _Traits >::is_open () [inline], [nodiscard]
```

Wrapper to test for an open file.

**Returns**

`rdbuf() -> is_open()`

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [iword\(\)](#).

Referenced by [iword\(\)](#).

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __c,
    char __default) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**open() [1/3]**

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path<_Path, void> std::basic_fstream<_CharT, _Traits>::open (
    const _Path & __s,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Opens an external file.

**Parameters**

|                     |                                                            |
|---------------------|------------------------------------------------------------|
| <code>__s</code>    | The name of the file, as a <code>filesystem::path</code> . |
| <code>__mode</code> | The open mode flags.                                       |

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

**open() [2/3]**

```
template<typename _CharT, typename _Traits>
void std::basic_fstream<_CharT, _Traits>::open (
```

```
const char * __s,
ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Opens an external file.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

#### **open()** [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_fstream< _CharT, _Traits >::open (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Opens an external file.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

#### **operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

#### **operator"!"()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

#### **operator<<()** [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<()** [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    __ostream_type & (* __pf) (__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

**operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.



**Parameters**

|       |                             |
|-------|-----------------------------|
| $\_p$ | A variable of pointer type. |
|-------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [6/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|       |                                            |
|-------|--------------------------------------------|
| $\_f$ | A variable of builtin floating point type. |
|-------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [7/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|       |                                            |
|-------|--------------------------------------------|
| $\_f$ | A variable of builtin floating point type. |
|-------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\leftarrow$   | A variable of builtin integral type. |
| $\leftarrow n$ |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(* __pf ) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "`std::cout << std::endl`". For more information, see the `io manip` header.

**operator<<()** [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\leftarrow$   | A variable of builtin integral type. |
| $\leftarrow n$ |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                |                                            |
|----------------|--------------------------------------------|
| $\leftarrow$   | A variable of builtin floating point type. |
| $\leftarrow$   |                                            |
| $\leftarrow$   |                                            |
| $\leftarrow$   |                                            |
| $\leftarrow f$ |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    short __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References `basic_ostream()`, `std::ios_base::badbit`, `std::ostreambuf_iterator<_CharT, _Traits>::failed()`, `std::ios_base::goodbit`, `std::num_put<_CharT, _OutTtr>::put()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::use_facet()`.

**operator<<()** [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    __istream_type &(* __pf) (__istream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
    __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n) [inherited]
```

Integer arithmetic extractors.

## Parameters

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>()** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    ios_base &(* __pf ) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|                                                                                       |                                            |
|---------------------------------------------------------------------------------------|--------------------------------------------|
| $\leftrightarrow$<br>$\leftrightarrow$<br>$\leftrightarrow$<br>$\leftrightarrow$<br>f | A variable of builtin floating point type. |
|---------------------------------------------------------------------------------------|--------------------------------------------|



**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

**operator>>()** [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

`*this`

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**putback()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                          |                       |
|--------------------------|-----------------------|
| $\leftrightarrow$<br>_ix | Index into the array. |
|--------------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_fstream< _CharT, _Traits >::rdbuf () const [inline], [nodiscard]
Accessing the underlying buffer.
```

**Returns**

The current basic\_filebuf buffer.

This hides both signatures of std::basic\_ios::rdbuf().

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|      |                        |
|------|------------------------|
| __sb | The new stream buffer. |
|------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument rdbuf(), which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios<char_type, traits_type>::bad()`, `std::basic_ios<char_type, traits_type>::clear()`, `std::basic_ios<char_type, traits_type>::eof()`, `std::basic_ios<char_type, traits_type>::fail()`, `std::basic_ios<char_type, traits_type>::good()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::seekg()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read (
    char_type * __s,
    streamsize __n) [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::readsome (
    char_type * __s,
    streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

References [fmtflags](#).

**seekg() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
    off_type __off,
    ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekg()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekp()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    off_type __off,
    ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekp()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    pos_type __pos) [inherited]
```

Changing the current write position.



**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets `failbit`.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf() [1/2]**

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf() [2/2]**

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
    iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

## Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::getline()`, `std::getline()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::ws()`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
    void ) [inherited]
```

Synchronizing the stream buffer.

## Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

## Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void ) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References [std::ios\\_base::badbit](#), [std::ios\\_base::cur](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::in](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#).

**tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp () [inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

References [std::ios\\_base::cur](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::out](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#).

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), and [copyfmt\(\)](#).

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::rdstate()`.

Referenced by `std::__detail::operator>>()`.

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References `fmtflags`.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::basic_ios< char_type, traits_type >::fill()`, `std::getline()`, `std::getline()`, `std::subtract_with_carry_engine< uint_fast32_t, std::tr2::operator>>()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::seed()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n) [inherited]
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

\*this

Characters are copied from `___s` and inserted into the stream until one of the following happens:

- `___n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.218.6 Member Data Documentation****`_M_gcount`**

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream<_CharT, _Traits>::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by [get\(\)](#), [get\(\)](#), [get\(\)](#), [get\(\)](#), [getline\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [putback\(\)](#), [read\(\)](#), [readsome\(\)](#), and [unget\(\)](#).

**`adjustfield`**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Referenced by [std::num\\_put<\\_CharT, \\_Outiter>::do\\_put\(\)](#), [std::internal\(\)](#), [std::left\(\)](#), and [std::right\(\)](#).

**`app`**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

**`ate`**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::open\(\)](#).

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char_type, traits_type >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< char_type, traits_type >::tellg()`, and `std::ws()`.

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`.

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<char_type, traits_type>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<char_type, traits_type>::seekg()`, and `std::ws()`.

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<char_type, traits_type>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, and `std::basic_ostream<_CharT, _Traits>::seekp()`.

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by `std::fixed()`, and `std::hexfloat()`.

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_ostream<_CharT, _Traits>::ignore()`.



`std::basic_ostream< char_type, traits_type >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::ws()` and `std::ws()`.

## hex

const `fmtflags` `std::ios_base::hex` [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::hex()`.

## in

const `openmode` `std::ios_base::in` [static], [inherited]

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::xsgetn()` and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

## internal

const `fmtflags` `std::ios_base::internal` [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by `std::internal()`.

## left

const `fmtflags` `std::ios_base::left` [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

## oct

const `fmtflags` `std::ios_base::oct` [static], [inherited]

Converts integer input or generates integer output in octal base.

Referenced by `std::oct()`.

## out

const `openmode` `std::ios_base::out` [static], [inherited]

Open for output. Default for `ofstream` and `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

## right

const `fmtflags` `std::ios_base::right` [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.) Referenced by [std::right\(\)](#).

### scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

### showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

### showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

### showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

### skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), and [std::skipws\(\)](#).

### trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

### unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

### uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following file:

- [fstream](#)

## 5.219 `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >` Class Template Reference

`#include <assoc_container.hpp>`

Inheritance diagram for `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >`:



### Protected Member Functions

- **`basic_hash_table`** (const [basic\\_hash\\_table](#) &other)
- `template<typename T0>`  
**`basic_hash_table`** (T0 t0)
- `template<typename T0, typename T1>`  
**`basic_hash_table`** (T0 t0, T1 t1)
- `template<typename T0, typename T1, typename T2>`  
**`basic_hash_table`** (T0 t0, T1 t1, T2 t2)
- `template<typename T0, typename T1, typename T2, typename T3>`  
**`basic_hash_table`** (T0 t0, T1 t1, T2 t2, T3 t3)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4>`  
**`basic_hash_table`** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5>`  
**`basic_hash_table`** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>`  
**`basic_hash_table`** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>`  
**`basic_hash_table`** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8>`  
**`basic_hash_table`** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

### 5.219.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename Resize_Policy,
bool Store_Hash, typename Tag, typename Policy_Tl, typename _Alloc>
class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag,
Policy_Tl, _Alloc >
```

A hashed container abstraction.

#### Template Parameters

|                      |                                                                      |
|----------------------|----------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                            |
| <i>Mapped</i>        | Map type.                                                            |
| <i>Hash_Fn</i>       | Hashing functor.                                                     |
| <i>Eq_Fn</i>         | Equal functor.                                                       |
| <i>Resize_Policy</i> | Resizes hash.                                                        |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. |
| <i>Tag</i>           | Instantiating data structure type, see container_tag.                |
| <i>Policy_Tl</i>     | Policy typelist.                                                     |
| <i>_Alloc</i>        | Allocator type.                                                      |

Base is dispatched at compile time via Tag, from the following choices: cc\_hash\_tag, gp\_hash\_tag, and descendants of basic\_hash\_tag.

Base choices are: detail::cc\_ht\_map, detail::gp\_ht\_map

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.220 \_\_gnu\_pbds::basic\_hash\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::basic_hash_tag`:



### 5.220.1 Detailed Description

Basic hash structure.

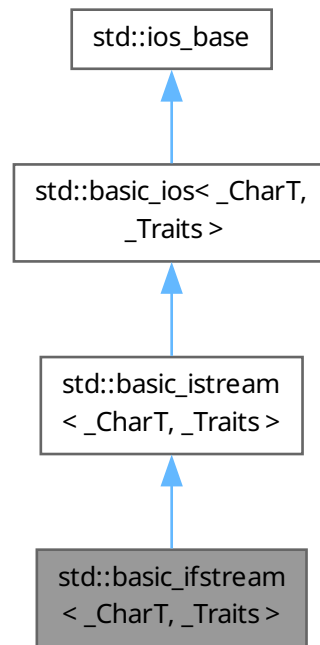
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.221 `std::basic_ifstream<_CharT, _Traits>` Class Template Reference

```
#include <fstream>
```

Inheritance diagram for std::basic\_ifstream< \_CharT, \_Traits >:



## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
  - typedef [basic\\_filebuf](#)< char\_type, traits\_type > **\_\_filebuf\_type**
  - typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
  - typedef [basic\\_istream](#)< char\_type, traits\_type > **\_\_istream\_type**
  - typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**
  - typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
  - typedef \_CharT **char\_type**
  - enum [event](#) { [erase\\_event](#) , [imbue\\_event](#) , [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
  - typedef traits\_type::int\_type **int\_type**
  - typedef \_ios\_ostate [iostate](#)
  - typedef traits\_type::off\_type **off\_type**
  - typedef \_ios\_Openmode [openmode](#)
  - typedef traits\_type::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir [seekdir](#)
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**

## Public Member Functions

- [basic\\_ifstream](#) ()
- **basic\_ifstream** ([basic\\_ifstream](#) &&\_\_rhs)
- template<typename \_Path, typename \_Require = \_If\_fs\_path<\_Path>>  
  [basic\\_ifstream](#) (const \_Path &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- **basic\_ifstream** (const [basic\\_ifstream](#) &)=delete
- [basic\\_ifstream](#) (const char \*\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_ifstream](#) (const [std::string](#) &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- ~[basic\\_ifstream](#) ()
- template<typename \_ValueT>  
  [basic\\_istream](#)<\_CharT, \_Traits> & **\_M\_extract** (\_ValueT &\_\_v)
- const [locale](#) & **\_M\_getloc** () const
- void **\_M\_setstate** ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize](#) [gcount](#) () const
- [basic\\_istream](#)<char> & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [basic\\_istream](#)<wchar\_t> & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [basic\\_istream](#)<char> & **ignore** ([streamsize](#) \_\_n)
- [basic\\_istream](#)<wchar\_t> & **ignore** ([streamsize](#) \_\_n)
- [basic\\_istream](#)<char> & **ignore** ([streamsize](#) \_\_n, int\_type \_\_delim)
- [basic\\_istream](#)<wchar\_t> & **ignore** ([streamsize](#) \_\_n, int\_type \_\_delim)
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- bool [is\\_open](#) ()
- bool **is\_open** () const
- long & [iword](#) (int \_\_ix)
- char [narrow](#) (char\_type \_\_c, char \_\_default) const
- template<typename \_Path>  
  \_If\_fs\_path<\_Path, void> [open](#) (const \_Path &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- void [open](#) (const char \*\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- void [open](#) (const [std::string](#) &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_ifstream](#) & **operator=** ([basic\\_ifstream](#) &&\_\_rhs)
- [basic\\_ifstream](#) & **operator=** (const [basic\\_ifstream](#) &)=delete
- \_\_istream\_type & **operator>>** (\_\_streambuf\_type \*\_\_sb)
- \_\_istream\_type & **operator>>** (void \*&\_\_p)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)

- `__filebuf_type * rdbuf () const`
- `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iosstate __state)`
- `void swap (basic_ifstream & __rhs)`
- `basic_ostream<_CharT, _Traits> * tie () const`
- `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __tiestr)`
- `void unsetf (fmtflags __mask)`
- `char_type widen (char __c) const`
- `streamsize width () const`
- `streamsize width (streamsize __wide)`
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
- `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool & __n)`
- `__istream_type & operator>> (short & __n)`
- `__istream_type & operator>> (unsigned short & __n)`
- `__istream_type & operator>> (int & __n)`
- `__istream_type & operator>> (unsigned int & __n)`
- `__istream_type & operator>> (long & __n)`
- `__istream_type & operator>> (unsigned long & __n)`
- `__istream_type & operator>> (long long & __n)`
- `__istream_type & operator>> (unsigned long long & __n)`
- `__istream_type & operator>> (float & __f)`
- `__istream_type & operator>> (double & __f)`
- `__istream_type & operator>> (long double & __f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.



- `int_type get ()`
  - `__istream_type & get (char_type &__c)`
  - `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
  - `__istream_type & get (char_type *__s, streamsize __n)`
  - `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
  - `__istream_type & get (__streambuf_type &__sb)`
  - `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `__istream_type & getline (char_type *__s, streamsize __n)`
  - `__istream_type & ignore (streamsize __n, int_type __delim)`
  - `__istream_type & ignore (streamsize __n)`
  - `__istream_type & ignore ()`
  - `int_type peek ()`
  - `__istream_type & read (char_type *__s, streamsize __n)`
  - `streamsize readsome (char_type *__s, streamsize __n)`
  - `__istream_type & putback (char_type __c)`
  - `__istream_type & unget ()`
  - `int sync ()`
  - `pos_type tellg ()`
  - `__istream_type & seekg (pos_type)`
  - `__istream_type & seekg (off_type, ios_base::seekdir)`
- 
- `operator bool () const`
  - `bool operator! () const`

#### Static Public Member Functions

- `static bool sync_with_stdio (bool __sync=true)`
- `static int xalloc () throw ()`

#### Public Attributes

- `class __attribute__((__abi_tag__("cxx11"))) failure typedef _los_Fmtflags fmtflags`

#### Static Public Attributes

- `static const openmode __noreplace`
- `static const fmtflags adjustfield`
- `static const openmode app`
- `static const openmode ate`
- `static const iostate badbit`
- `static const fmtflags basefield`
- `static const seekdir beg`
- `static const openmode binary`
- `static const fmtflags boolalpha`
- `static const seekdir cur`
- `static const fmtflags dec`
- `static const seekdir end`
- `static const iostate eofbit`
- `static const iostate failbit`
- `static const fmtflags fixed`
- `static const fmtflags floatfield`
- `static const iostate goodbit`
- `static const fmtflags hex`
- `static const openmode in`

- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const `locale` & \_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT>  
`__istream_type` & `_M_extract` (\_ValueT & \_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `_M_move` (`ios_base` &) noexcept
- void `_M_swap` (`ios_base` & \_\_rhs) noexcept
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \* \_\_sb)
- void `move` (`basic_ios` && \_\_rhs)
- void `move` (`basic_ios` & \_\_rhs)
- void `set_rdbuf` (`basic_streambuf`< `_CharT`, `_Traits` > \* \_\_sb)
- void `swap` (`basic_ios` & \_\_rhs) noexcept
- void `swap` (`basic_istream` & \_\_rhs)

### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const `__ctype_type` \* `_M_ctype`
- `iosstate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `streamsize` `_M_gcount`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` \* `_M_num_get`
- const `__num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > \* `_M_streambuf`
- `iosstate` `_M_streambuf_state`

- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

### 5.221.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ifstream<_CharT, _Traits>
```

Controlling input for files.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

### 5.221.2 Member Typedef Documentation

#### `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

#### `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

#### `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`

- eofbit
- failbit
- goodbit

### openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

### seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

## 5.221.3 Member Enumeration Documentation

### event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

## 5.221.4 Constructor & Destructor Documentation

### `basic_ifstream()` [1/4]

```
template<typename _CharT, typename _Traits>
std::basic_ifstream<_CharT, _Traits>::basic_ifstream () [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

**basic\_ifstream()** [2/4]

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
    const char * __s,
    ios_base::openmode __mode = ios_base::in) [inline], [explicit]
```

Create an input file stream.

**Parameters**

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::in` is automatically included in `__mode`.

**basic\_ifstream()** [3/4]

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::in) [inline], [explicit]
```

Create an input file stream.

**Parameters**

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | <code>std::string</code> specifying the filename.              |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::in` is automatically included in `__mode`.

**basic\_ifstream()** [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
    const _Path & __s,
    ios_base::openmode __mode = ios_base::in) [inline]
```

Create an input file stream.

**Parameters**

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | <code>filesystem::path</code> specifying the filename.         |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::in` is automatically included in `__mode`.

**~basic\_ifstream()**

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::~~basic_ifstream () [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

### 5.221.5 Member Function Documentation

#### `_M_getloc()`

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_date\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_weekday\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_year\(\)](#), [std::num\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), [std::time\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::get\(\)](#), and [std::time\\_put<\\_CharT, \\_Outlter>::put\(\)](#).

#### `bad()`

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic\_ios<\_CharT, \_Traits>::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

#### Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

#### `clear()`

```
template<typename _CharT, typename _Traits>
```

```
void std::basic\_ios<\_CharT, \_Traits>::clear (  
    iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

#### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::exceptions\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::unget\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

#### `close()`

```
template<typename _CharT, typename _Traits>
```

```
void std::basic\_ifstream<\_CharT, \_Traits>::close () [inline]
```

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of \_\_rhs into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof() const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::exceptions() const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::exceptions (
    iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
```



```
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

### fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
Fast error checking.
```

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::regex\\_traits< \\_Ch\\_type >::value\(\)](#).

### fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
Retrieves the empty character.
```

#### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by [copyfmt\(\)](#), [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#), and [std::subtract\\_with\\_carry\\_engine](#).

### fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.



**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References [\\_M\\_gcount](#), and [std::ios\\_base::goodbit](#).

**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c) [inherited]
```

Simple extraction.

## Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

## Returns

`*this`

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

## Note

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::get (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

## Parameters

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

## Returns

`*this`

Returns `get(__s, __n, widen('\n'))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Simple multiple-character extraction.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>](#):

**getline()** [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s,__n,widen("\n"))`.

**getline()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References `_M_gcount`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**getline()** [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale::global C++ locale`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outiter>::do_put()`, `std::basic_ios<_CharT, _Traits>::do_get()`, `std::chrono::operator<<()`, `std::operator>>()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::__detail::operator>>()`.

**ignore()** [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
    void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#).

**ignore()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::rdbuf\(\)](#).

**ignore()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim) [inherited]
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::rdbuf\(\)](#).

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References `std::ios_base::getloc()`, `std::ios_base::imbue()`, and `rdbuf()`.

Referenced by `std::chrono::operator<<()`.

**init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
    basic_streambuf<_CharT, _Traits> * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_ios<char_type, traits_type>::basic_ios()`, and `std::basic_ostream<char_type, traits_type>::basic_ostream()`.

**is\_open()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ifstream<_CharT, _Traits>::is_open () [inline], [nodiscard]
```

Wrapper to test for an open file.

**Returns**

`rdbuf()->is_open()`

**word()**

```
long & std::ios_base::word (
    int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `word` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References `word()`.

Referenced by `word()`.



**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
    char_type __c,
    char __default) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**open() [1/3]**

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, void > std::basic_ifstream< _CharT, _Traits >::open (
    const _Path & __s,
    ios_base::openmode __mode = ios_base::in) [inline]
```

Opens an external file.

**Parameters**

|                     |                                                            |
|---------------------|------------------------------------------------------------|
| <code>__s</code>    | The name of the file, as a <code>filesystem::path</code> . |
| <code>__mode</code> | The open mode flags.                                       |

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

**open() [2/3]**

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream< _CharT, _Traits >::open (
    const char * __s,
    ios_base::openmode __mode = ios_base::in) [inline]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

**open()** [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream<_CharT, _Traits>::open (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::in) [inline]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
    __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

**operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
    __istream_type &(* __pf) (__istream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

**operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::operator>> (
    __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                |                                            |
|----------------|--------------------------------------------|
| <code>↵</code> | A variable of builtin floating point type. |
| <code>↵</code> |                                            |
| <code>↵</code> |                                            |
| <code>↵</code> |                                            |
| <code>f</code> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
    int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| <code>↵</code> | A variable of builtin integral type. |
| <code>n</code> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>()** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    ios_base &(* __pf ) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↔</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                                                                                          |                                            |
|------------------------------------------------------------------------------------------|--------------------------------------------|
| <code>↔</code><br><code>_↔</code><br><code>↔</code><br><code>_↔</code><br><code>f</code> | A variable of builtin floating point type. |
|------------------------------------------------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↔</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    short & __n) [inherited]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>_↔</code>  | A variable of builtin integral type. |
| <code>__n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data. References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

**operator>>()** [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
    unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>_↔</code>  | A variable of builtin integral type. |
| <code>__n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
    unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>_↔</code>  | A variable of builtin integral type. |
| <code>__n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
    unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↔</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [16/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↔</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [17/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                                    |                             |
|------------------------------------|-----------------------------|
| <code>_↔</code><br><code>_p</code> | A variable of pointer type. |
|------------------------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**putback()**

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::putback (
    char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.



### Parameters

|                          |                       |
|--------------------------|-----------------------|
| $\leftrightarrow$<br>_ix | Index into the array. |
|--------------------------|-----------------------|

### Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

### rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_ifstream< _CharT, _Traits >::rdbuf () const [inline], [nodiscard]
Accessing the underlying buffer.
```

### Returns

The current basic\_filebuf buffer.

This hides both signatures of std::basic\_ios::rdbuf().

### rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

### Parameters

|      |                        |
|------|------------------------|
| __sb | The new stream buffer. |
|------|------------------------|

### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument rdbuf(), which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

References [clear\(\)](#).

### rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios<char_type, traits_type>::bad()`, `clear()`, `std::basic_ios<char_type, traits_type>::eof()`,

`std::basic_ios<char_type, traits_type>::fail()`, `std::basic_ios<char_type, traits_type>::good()`, `std::basic_istream<_CharT, _Traits>::`

`std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::`

and `std::basic_istream<_CharT, _Traits>::unget()`.

**read()**

```
template<typename _CharT, typename _Traits>
```

```
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read (
    char_type * __s,
    streamsize __n) [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`,

`std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**readsome()**

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream<_CharT, _Traits>::readsome (
    char_type * __s,
    streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

References [fmtflags](#).

**seekg() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
    off_type __off,
    ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekg()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::seekg (
    pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::getline()`, `std::getline()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::ws()`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
    void ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pubsync\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_ifstream< _CharT, _Traits >::pos_type std::basic_ifstream< _CharT, _Traits >::tellg (
    void ) [inherited]
```

Getting the current read position.

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

## Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `copyfmt()`.

**tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

## Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References [fmtflags](#).

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::basic_ios<char_type, traits_type>::fill()`, `std::getline()`, `std::getline()`, `std::subtract_with_carry_engine<uint_fast32_t, std::tr2::operator>>()`, and `std::subtract_with_carry_engine<uint_fast32_t, 24, 10, 24>::seed()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.



## 5.221.6 Member Data Documentation

### **`_M_gcount`**

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by `get()`, `get()`, `get()`, `get()`, `getline()`, `ignore()`, `ignore()`, `ignore()`, `putback()`, `read()`, `readsome()`, and `unget()`.

### **`adjustfield`**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

### **`app`**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

### **`ate`**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

### **`badbit`**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char_type, traits_type >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::clear()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_ostream< char_type, traits_type >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< char_type, traits_type >::ws()`, and `std::ws()`.

### **`basefield`**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

### **`beg`**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↔filestreams.binary>.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`.

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<char_type, traits_type>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<char_type, traits_type>::seekg()`, and `std::ws()`.

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, and `std::time_get<_CharT, _InIter>::do_get_weekday()`.

`std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< char_type, traits_type >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::seekp()`.

## fixed

const `fmtflags` `std::ios_base::fixed` [static], [inherited]

Generate floating-point output in fixed-point notation.

Referenced by `std::fixed()`, and `std::hexfloat()`.

## floatfield

const `fmtflags` `std::ios_base::floatfield` [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

## goodbit

const `iosstate` `std::ios_base::goodbit` [static], [inherited]

Indicates all is well.

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::ws()`.

## hex

const `fmtflags` `std::ios_base::hex` [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::hex()`.

## in

const `openmode` `std::ios_base::in` [static], [inherited]

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::xsgetn()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

## internal

const `fmtflags` `std::ios_base::internal` [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

### left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by [std::num\\_put<\\_CharT, \\_Outiter>::do\\_put\(\)](#), and [std::left\(\)](#).

### oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

### out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

### right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Referenced by [std::right\(\)](#).

### scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

### showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put<\\_CharT, \\_Outiter>::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

### showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

### showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), and [std::skipws\(\)](#).

**trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

**unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following file:

- [fstream](#)

**5.222 \_\_gnu\_pbds::basic\_invalidation\_guarantee Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::basic_invalidation_guarantee`:



### 5.222.1 Detailed Description

The documentation for this struct was generated from the following file:

## 5.223 std::basic\_ios<\_CharT, \_Traits> Class Template Reference

Inheritance diagram for `std::basic_ios<_CharT, _Traits>`:

## Public Types

- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
- typedef [\\_ios\\_istate](#) [istate](#)
- typedef [\\_ios\\_Openmode](#) [openmode](#)
- typedef [\\_ios\\_Seekdir](#) [seekdir](#)
  
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits::int\\_type](#) [int\\_type](#)
- typedef [\\_Traits::pos\\_type](#) [pos\\_type](#)
- typedef [\\_Traits::off\\_type](#) [off\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
  
- typedef [ctype](#)< [\\_CharT](#) > [\\_\\_ctype\\_type](#)
- typedef [num\\_put](#)< [\\_CharT](#), [ostreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_put\\_type](#)
- typedef [num\\_get](#)< [\\_CharT](#), [istreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_get\\_type](#)

### Public Member Functions

- [basic\\_ios](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- virtual [~basic\\_ios](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([istate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([istate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [istate](#) [exceptions](#) () const
- void [exceptions](#) ([istate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) & \_\_loc)
- long & [iword](#) (int \_\_ix)
- char [narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) () const
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- [istate](#) [rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [setstate](#) ([istate](#) \_\_state)
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) () const
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) ([basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_tiestr)

- void `unsetf` (`fmtflags` \_\_mask)
- `char_type` `widen` (char \_\_c) const
- `streamsize` `width` () const
- `streamsize` `width` (`streamsize` \_\_wide)
- `operator bool` () const
- bool `operator!` () const

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Public Attributes

- class \_\_attribute\_\_((\_\_abi\_tag\_\_("cxx11"))) failure typedef `_ios_Fmtflags` `fmtflags`

### Static Public Attributes

- static const `openmode` `__noreplace`
- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`



## Protected Types

- enum { **\_S\_local\_word\_size** }

## Protected Member Functions

- **basic\_ios** ()
- **basic\_ios** (const **basic\_ios** &)=delete
- void **\_M\_cache\_locale** (const **locale** &\_\_loc)
- void **\_M\_call\_callbacks** (**event** \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- **\_Words** & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- void **\_M\_move** (**ios\_base** &) noexcept
- void **\_M\_swap** (**ios\_base** &\_\_rhs) noexcept
- void **init** (**basic\_streambuf**< **\_CharT**, **\_Traits** > \*\_\_sb)
- void **move** (**basic\_ios** &&\_\_rhs)
- void **move** (**basic\_ios** &\_\_rhs)
- **basic\_ios** & **operator=** (const **basic\_ios** &)=delete
- void **set\_rdbuf** (**basic\_streambuf**< **\_CharT**, **\_Traits** > \*\_\_sb)
- void **swap** (**basic\_ios** &\_\_rhs) noexcept

## Protected Attributes

- **\_Callback\_list** \* **\_M\_callbacks**
- const **\_\_ctype\_type** \* **\_M\_ctype**
- **iostate** **\_M\_exception**
- **char\_type** **\_M\_fill**
- bool **\_M\_fill\_init**
- **fmtflags** **\_M\_flags**
- **locale** **\_M\_ios\_locale**
- **\_Words** **\_M\_local\_word** [**\_S\_local\_word\_size**]
- const **\_\_num\_get\_type** \* **\_M\_num\_get**
- const **\_\_num\_put\_type** \* **\_M\_num\_put**
- **streamsize** **\_M\_precision**
- **basic\_streambuf**< **\_CharT**, **\_Traits** > \* **\_M\_streambuf**
- **iostate** **\_M\_streambuf\_state**
- **basic\_ostream**< **\_CharT**, **\_Traits** > \* **\_M\_tie**
- **streamsize** **\_M\_width**
- **\_Words** \* **\_M\_word**
- int **\_M\_word\_size**
- **\_Words** **\_M\_word\_zero**

### 5.223.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ios< _CharT, _Traits >
```

Template class **basic\_ios**, virtual base class for all stream classes.

#### Template Parameters

|               |                           |
|---------------|---------------------------|
| <b>_CharT</b> | Type of character stream. |
|---------------|---------------------------|

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
|----------------------|---------------------------------------------------------------------------------|

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

### 5.223.2 Member Typedef Documentation

#### `__ctype_type`

```
template<typename _CharT, typename _Traits>
typedef __ctype_type<_CharT> std::basic_ios<_CharT, _Traits>::__ctype_type
```

These are non-standard types.

#### `__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type
```

These are non-standard types.

#### `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type
```

These are non-standard types.

#### `char_type`

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_ios<_CharT, _Traits>::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

#### `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

### int\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::basic_ios< _CharT, _Traits >::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

### iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

### off\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::off_type std::basic_ios< _CharT, _Traits >::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

### openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

### pos\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

**traits\_type**

```
template<typename _CharT, typename _Traits>
```

```
typedef _Traits std::basic_ios<_CharT, _Traits>::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**5.223.3 Member Enumeration Documentation****event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

**5.223.4 Constructor & Destructor Documentation****basic\_ios() [1/2]**

```
template<typename _CharT, typename _Traits>
```

```
std::basic_ios<_CharT, _Traits>::basic_ios (
    basic_streambuf<_CharT, _Traits> * __sb) [inline], [explicit]
```

Constructor performs initialization.

The parameter is passed by derived streams.

Referenced by `copyfmt()`.

**~basic\_ios()**

```
template<typename _CharT, typename _Traits>
```

```
virtual std::basic_ios<_CharT, _Traits>::~~basic_ios () [inline], [virtual]
```

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by `rdbuf()`.

**basic\_ios() [2/2]**

```
template<typename _CharT, typename _Traits>
```

```
std::basic_ios<_CharT, _Traits>::basic_ios () [inline], [protected]
```

Empty.

The default constructor does nothing and is not normally accessible to users.

### 5.223.5 Member Function Documentation

#### **`_M_getloc()`**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_date\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_weekday\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_year\(\)](#), [std::num\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), [std::time\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), and [std::time\\_put<\\_CharT, \\_Outlter>::put\(\)](#).

#### **`bad()`**

```
template<typename _CharT, typename _Traits>  
bool std::basic\_ios<\_CharT, \_Traits>::bad () const [inline], [nodiscard]
```

Fast error checking.

##### Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

#### **`clear()`**

```
template<typename _CharT, typename _Traits>  
void std::basic\_ios<\_CharT, \_Traits>::clear (  
    iostate __state = goodbit)
```

[Re]sets the error state.

##### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::exceptions\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::unget\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

#### **`copyfmt()`**

```
template<typename _CharT, typename _Traits>  
basic\_ios<\_CharT, \_Traits> & std::basic\_ios<\_CharT, \_Traits>::copyfmt (  
    const basic\_ios<\_CharT, \_Traits> & __rhs)
```

Copies fields of `__rhs` into this.

##### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except) [inline]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
```

```
f.setstate (std::ios_base::badbit);

std::cerr << "Setting exception mask\n";
f.exceptions (std::ios_base::badbit);
}
```

### fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard]
Fast error checking.
```

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#), and [std::regex\\_traits< \\_Ch\\_type >::value\(\)](#).

### fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard]
Retrieves the empty character.
```

#### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by [copyfmt\(\)](#), [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#), and [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#).

### fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch) [inline]
Sets a new empty character.
```

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

### flags() [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
Access to format flags.
```

**Returns**

The format control flags for both input and output.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t >::do\\_subtract\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator<<\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t >::do\\_subtract\(\)](#).

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

References [fmtflags](#).

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale::global()`, the global C++ locale.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::money\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::do\\_get\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc)
```

Moves to a new locale.



**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References `std::ios_base::getloc()`, `std::ios_base::imbue()`, and `rdbuf()`.

Referenced by `std::chrono::operator<<()`.

**init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
    basic_streambuf<_CharT, _Traits> * __sb) [protected]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_ios<char_type, traits_type>::basic_ios()`, and `std::basic_ostream<char_type, traits_type>::basic_ostream()`.

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References `iword()`.

Referenced by `iword()`.

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __C,
    char __dfault) const [inline]
```

Squeezes characters.

## Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

## Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
```

```
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator"!()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

## Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

## Returns

The previous value of `precision()`.

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Referenced by [clear\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::getline\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [imbue\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellp\(\)](#), and [std::ws\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb)
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdstate()**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios< char_type, traits_type >::bad()`, `clear()`, `std::basic_ios< char_type, traits_type >::eof()`,

`std::basic_ios< char_type, traits_type >::fail()`, `std::basic_ios< char_type, traits_type >::good()`, `std::basic_istream< _CharT, _Traits >::`

`std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type >::`

and `std::basic_istream< _CharT, _Traits >::unget()`.

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

References [fmtflags](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state) [inline]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), and [std::ws\(\)](#).

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard]
```

Fetches the current *tied* stream.

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `copyfmt()`.

**tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr) [inline]
```

Ties this stream to an output stream.

## Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

## Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References `fmtflags`.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::basic_ios< char_type, traits_type >::fill()`, `std::getline()`, `std::getline()`, `std::subtract_with_carry_engine< uint_fast32_t, std::tr2::operator>>()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::seed()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.223.6 Member Data Documentation****adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::internal\(\)](#), [std::left\(\)](#), and [std::right\(\)](#).

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#).

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#).

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~sentry\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::clear\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_ostream< char\\_type, traits\\_type >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::tellg\(\)](#), and [std::ws\(\)](#).

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Referenced by [std::dec\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::hex\(\)](#), and [std::oct\(\)](#).

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekpos\(\)](#).



## binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↔filestreams.binary>.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::showmanyc\(\)](#).

## boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

## cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::imbue\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), and [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#).

## dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#).

## end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::open\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#).

## eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_date\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_time\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::read\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::read\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::seekg\(\)](#), and [std::ws\(\)](#).

## failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), and [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_weekday\(\)](#).

std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< char\_type, traits\_type >::get(), std::time\_get< \_CharT, \_Inlter >::get(), std::basic\_istream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::seekp(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

### fixed

const `fmtflags` std::ios\_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

### floatfield

const `fmtflags` std::ios\_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

### goodbit

const `iosstate` std::ios\_base::goodbit [static], [inherited]

Indicates all is well.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::num\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_year\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::getline\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::readsomewg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ws\(\)](#), and [std::ws\(\)](#).

### hex

const `fmtflags` std::ios\_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Referenced by [std::num\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), and [std::hex\(\)](#).

### in

const `openmode` std::ios\_base::in [static], [inherited]

Open for input. Default for `ifstream` and `fstream`.

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::xsgetn\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

### internal

const `fmtflags` std::ios\_base::internal [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

### **left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::left\(\)](#).

### **oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

### **out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

### **right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Referenced by [std::right\(\)](#).

### **scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

### **showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

### **showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

### **showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), and [std::skipws\(\)](#).

**trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

**unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put< \\_CharT, \\_Outiter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [basic\\_ios.h](#)
- [basic\\_ios.tcc](#)

**5.224 std::basic\_istream< \_CharT, \_Traits > Class Template Reference**

```
#include <istream>
```

Inheritance diagram for [std::basic\\_istream< \\_CharT, \\_Traits >](#):

**Public Types**

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
- typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**

- typedef [basic\\_ios](#)< [\\_CharT](#), [\\_Traits](#) > [\\_\\_ios\\_type](#)
- typedef [basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > [\\_\\_istream\\_type](#)
- typedef [num\\_get](#)< [\\_CharT](#), [istreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_get\\_type](#)
- typedef [num\\_put](#)< [\\_CharT](#), [ostreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_put\\_type](#)
- typedef [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > [\\_\\_ostream\\_type](#)
- typedef [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > [\\_\\_streambuf\\_type](#)
- typedef [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > [\\_\\_streambuf\\_type](#)
- typedef [\\_CharT](#) [char\\_type](#)
- enum [event](#) { [erase\\_event](#) , [imbue\\_event](#) , [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
- typedef [\\_Traits::int\\_type](#) [int\\_type](#)
- typedef [\\_ios::iostate](#) [iostate](#)
- typedef [\\_Traits::off\\_type](#) [off\\_type](#)
- typedef [\\_ios::Openmode](#) [openmode](#)
- typedef [\\_Traits::pos\\_type](#) [pos\\_type](#)
- typedef [\\_ios::Seekdir](#) [seekdir](#)
- typedef [\\_Traits](#) [traits\\_type](#)

## Public Member Functions

- [basic\\_istream](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- virtual [~basic\\_istream](#) ()
- template<typename [\\_ValueT](#)>  
[basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > & [\\_M\\_extract](#) ([\\_ValueT](#) & \_\_v)
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename [\\_ValueT](#)>  
[basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > & [\\_M\\_insert](#) ([\\_ValueT](#) \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [streamsize](#) [gcount](#) () const
- [basic\\_istream](#)< [char](#) > & [getline](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [basic\\_istream](#)< [wchar\\_t](#) > & [getline](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [locale](#) [imbue](#) (const [locale](#) & \_\_loc)
- long & [iword](#) (int \_\_ix)

- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (const void \*\_\_p)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (nullptr\_t)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- [\\_\\_istream\\_type](#) & [operator>>](#) (void \*&\_\_p)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) () const
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- [iostate](#) [rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [\\_\\_ostream\\_type](#) & [seekp](#) ([off\\_type](#), [ios\\_base::seekdir](#))
- [\\_\\_ostream\\_type](#) & [seekp](#) ([pos\\_type](#))
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [setstate](#) ([iostate](#) \_\_state)
- [pos\\_type](#) [telp](#) ()
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \*\_\_tiestr)
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- char\_type [widen](#) (char \_\_c) const
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_istream\\_type](#) &(\*\_\_pf)(\_\_istream\_type &))
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)(\_\_ios\_type &))
- [\\_\\_istream\\_type](#) & [operator>>](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Extractors

All the *operator>>* functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- [\\_\\_istream\\_type](#) & [operator>>](#) (bool &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (short &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned short &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (int &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned int &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (float &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (double &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long double &\_\_f)

## Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type & __sb, char_type __delim)`
- `__istream_type & get (__streambuf_type & __sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

## Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`

- `__ostream_type` & `operator<<` (float \_\_f)
- `__ostream_type` & `operator<<` (long double \_\_f)

### Unformatted Output Functions

*All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.*

*If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.*

*If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.*

- `__ostream_type` & `put` (char\_type \_\_c)
- `__ostream_type` & `write` (const char\_type \*\_\_s, streamsize \_\_n)
- `operator bool` () const
- `bool operator!` () const

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Public Attributes

- class `__attribute__((__abi_tag__("cxx11")))` failure typedef `_los_Fmtflags` `fmtflags`

### Static Public Attributes

- static const `openmode` `__noreplace`
- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`



- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- [basic\\_iostream](#) ([basic\\_iostream](#) && \_\_rhs)
- [basic\\_iostream](#) (const [basic\\_iostream](#) &)=delete
- void [\\_M\\_cache\\_locale](#) (const [locale](#) & \_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- template<typename \_ValueT>  
  [\\_\\_istream\\_type](#) & [\\_M\\_extract](#) (\_ValueT & \_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- template<typename \_ValueT>  
  [\\_\\_ostream\\_type](#) & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) & \_\_rhs) noexcept
- void [init](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- void [move](#) ([basic\\_ios](#) && \_\_rhs)
- void [move](#) ([basic\\_ios](#) & \_\_rhs)
- [basic\\_iostream](#) & [operator=](#) ([basic\\_iostream](#) && \_\_rhs)
- [basic\\_iostream](#) & [operator=](#) (const [basic\\_iostream](#) &)=delete
- void [set\\_rdbuf](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- void [swap](#) ([basic\\_ios](#) & \_\_rhs) noexcept
- void [swap](#) ([basic\\_iostream](#) & \_\_rhs)
- void [swap](#) ([basic\\_istream](#) & \_\_rhs)
- void [swap](#) ([basic\\_ostream](#) & \_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const [\\_\\_ctype\\_type](#) \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- [char\\_type](#) [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [streamsize](#) [\\_M\\_gcount](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* [\\_M\\_num\\_get](#)
- const [\\_\\_num\\_put\\_type](#) \* [\\_M\\_num\\_put](#)

- [streamsize \\_M\\_precision](#)
- [basic\\_streambuf< \\_CharT, \\_Traits > \\* \\_M\\_streambuf](#)
- [iostate \\_M\\_streambuf\\_state](#)
- [basic\\_ostream< \\_CharT, \\_Traits > \\* \\_M\\_tie](#)
- [streamsize \\_M\\_width](#)
- [\\_Words \\* \\_M\\_word](#)
- [int \\_M\\_word\\_size](#)
- [\\_Words \\_M\\_word\\_zero](#)

### 5.224.1 Detailed Description

**template**<typename [\\_CharT](#), typename [\\_Traits](#)>  
**class** [std::basic\\_iostream< \\_CharT, \\_Traits >](#)

Template class [basic\\_iostream](#).

Template Parameters

|                         |                                                                                    |
|-------------------------|------------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                          |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits&lt;_CharT&gt;</a> . |

This class multiply inherits from the input and output stream classes simply to provide a single interface.

### 5.224.2 Member Typedef Documentation

#### [event\\_callback](#)

`typedef void(* std::ios\_base::event\_callback) (event __e, ios\_base &__b, int __i) [inherited]`

The type of an event callback function.

Parameters

|                           |                                                        |
|---------------------------|--------------------------------------------------------|
| <a href="#">_↵<br/>_e</a> | One of the members of the event enum.                  |
| <a href="#">_↵<br/>_b</a> | Reference to the <a href="#">ios_base</a> object.      |
| <a href="#">_↵<br/>_i</a> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several [ios\\_base](#) and [basic\\_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

#### [iostate](#)

`typedef \_Ios\_Iostate std::ios\_base::iostate [inherited]`

This is a bitmask type.

[\\_Ios\\_Iostate](#) is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type [iostate](#) are:

- [badbit](#)
- [eofbit](#)
- [failbit](#)
- [goodbit](#)

### openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

### seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

## 5.224.3 Member Enumeration Documentation

### event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

## 5.224.4 Constructor & Destructor Documentation

### basic\_iostream()

```
template<typename _CharT, typename _Traits>  
std::basic_iostream< _CharT, _Traits >::basic_iostream (  
    basic_streambuf< _CharT, _Traits > * __sb) [inline], [explicit]
```

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

### ~basic\_iostream()

```
template<typename _CharT, typename _Traits>  
virtual std::basic_iostream< _CharT, _Traits >::~~basic_iostream () [inline], [virtual]
```

Destructor does nothing.

## 5.224.5 Member Function Documentation

**`_M_getloc()`**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_get< _CharT, _Inlter >::do_get_year()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

**`bad()`**

```
template<typename _CharT, typename _Traits>
bool std::basic\_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**`clear()`**

```
template<typename _CharT, typename _Traits>
void std::basic\_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios< char_type, traits_type >::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream< _CharT, _Traits >::rdbuf()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type >::unget()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**`copyfmt()`**

```
template<typename _CharT, typename _Traits>
basic\_ios< _CharT, _Traits > & std::basic\_ios< _CharT, _Traits >::copyfmt (
    const basic\_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

### eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

### Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

### exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

### exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
```

```

    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}

```

**fail()**

```

template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
Fast error checking.

```

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::regex\\_traits< \\_Ch\\_type >::value\(\)](#).

**fill() [1/2]**

```

template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
Retrieves the empty character.

```

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Referenced by [copyfmt\(\)](#), [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#), and [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#).

**fill() [2/2]**

```

template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch) [inline], [inherited]

```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

**flags() [1/2]**

```

fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]

```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::subtract\\_with\\_carry\\_engine<uint\\_fast32\\_t>::do\\_put\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator<<\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::subtract\\_with\\_carry\\_engine<uint\\_fast32\\_t>::do\\_put\(\)](#).

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

References [fmtflags](#).

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::pubsync\(\)](#).

**gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

`*this`

Returns `get(__sb,widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

`*this`

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References `_M_gcount`, and `std::ios_base::goodbit`.

**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c) [inherited]
```

Simple extraction.



**Parameters**

|                                                                                                                      |                                       |
|----------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| <a href="#"></a><br><code>_c</code> | The character in which to store data. |
|----------------------------------------------------------------------------------------------------------------------|---------------------------------------|

**Returns**

`*this`

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.



References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                                                                                                                         |                                                           |
|-------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| <a href="#"></a><br><code>__s</code> | Pointer to an array.                                      |
| <a href="#"></a><br><code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

`*this`

Returns `get(__s,__n,widen('\n'))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::](#)

**getline()** [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s,__n,widen("\n"))`.

**getline()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References [\\_M\\_gcount](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sgetc\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::snextc\(\)](#).

**getline()** [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::money\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::do\\_get\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

**ignore()** [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#).

**ignore()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::rdbuf\(\)](#).

**ignore()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim) [inherited]
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

`*this`

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::rdbuf\(\)](#).

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References `std::ios_base::getloc()`, `std::ios_base::imbue()`, and `rdbuf()`.

Referenced by `std::chrono::operator<<()`.

**init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
    basic_streambuf<_CharT, _Traits> * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_ios<char_type, traits_type>::basic_ios()`, and `std::basic_ostream<char_type, traits_type>::basic_ostream()`.

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References `iword()`.

Referenced by `iword()`.

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __C,
    char __default) const [inline], [inherited]
```

Squeezes characters.

## Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

## Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator"!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(* __pf )(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(* __pf )(__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

**operator<<() [4/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [5/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    double __f) [inline], [inherited]
```

Floating point arithmetic inserters.



**Parameters**

|                                                                                   |                                            |
|-----------------------------------------------------------------------------------|--------------------------------------------|
|  | A variable of builtin floating point type. |
|  |                                            |
|  |                                            |
|  |                                            |
| <i>f</i>                                                                          |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [7/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                                                                                     |                                            |
|-------------------------------------------------------------------------------------|--------------------------------------------|
|    | A variable of builtin floating point type. |
|    |                                            |
|   |                                            |
|  |                                            |
| <i>f</i>                                                                            |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                     |                                      |
|-------------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <i>n</i>                                                                            |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<()** [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    short __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put< \\_CharT, \\_Outiter >::put\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<()** [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

**operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type & (* __pf ) ( __istream_type & ) ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    __streambuf_type * __sb ) [inherited]
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| <i>f</i>        |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| <i>f</i>        |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [7/17]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>()** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.



**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [12/17]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

**operator>>() [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [14/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_↔$ | A variable of builtin integral type. |
| $\_n$ |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_↔$ | A variable of builtin integral type. |
| $\_n$ |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_↔$ | A variable of builtin integral type. |
| $\_n$ |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                        |                             |
|------------------------|-----------------------------|
| <code>_↔<br/>_p</code> | A variable of pointer type. |
|------------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision() [1/2]**

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::chrono::operator<<()`.

**precision() [2/2]**

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c) [inherited]
```

Simple insertion.

## Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

## Returns

`*this`

Tries to insert `__c`.

## Note

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits](#)

**putback()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c) [inherited]
```

Unextracting a single character.

## Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

## Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

## Note

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdstate\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::sputbackc\(\)](#).

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.

## Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

## Returns

A reference to a `void*` associated with the index.

The `pwd` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

## `rdbuf()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

## Returns

The current stream buffer.

This does not change the state of the stream.

Referenced by `clear()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `imbue()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellp()`, and `std::ws()`.

## `rdbuf()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

## Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

## Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

References `clear()`.

**rdstate()**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios<_CharT, _Traits>::rdstate() const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios<char_type, traits_type>::bad()`, `clear()`, `std::basic_ios<char_type, traits_type>::eof()`,

`std::basic_ios<char_type, traits_type>::fail()`, `std::basic_ios<char_type, traits_type>::good()`, `std::basic_istream<_CharT, _Traits>::`

`std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::`

and `std::basic_istream<_CharT, _Traits>::unget()`.

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read (
    char_type * __s,
    streamsize __n) [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

`*this`

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::readsome (
    char_type * __s,
    streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rddbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

References [fmtflags](#).

**seekg() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
    off_type __off,
    ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#) and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**seekg()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#) and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**seekp()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    off_type __off,
    ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).



**seekp()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    pos_type __pos) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets `failbit`.

References `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References `fmtflags`.

Referenced by `std::__detail::operator>>()`.

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl & mask`. An example mask is `ios_base::adjustfield`.

References `fmtflags`.

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See std::ios\_base::iostate for the possible bit values.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), and [std::ws\(\)](#).

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
    void ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pubsync\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void ) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References [std::ios\\_base::badbit](#), [std::ios\\_base::cur](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::in](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#).

**tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp () [inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

References [std::ios\\_base::cur](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::out](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#).

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), and [copyfmt\(\)](#).

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::rdstate()`.

Referenced by `std::__detail::operator>>()`.

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References `fmtflags`.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::basic_ios< char_type, traits_type >::fill()`, `std::getline()`, `std::getline()`, `std::subtract_with_carry_engine< uint_fast32_t, std::tr2::operator>>()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::seed()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n) [inherited]
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

\*this

Characters are copied from `___s` and inserted into the stream until one of the following happens:

- `___n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( )    [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.224.6 Member Data Documentation****`_M_gcount`**

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount    [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by `get()`, `get()`, `get()`, `get()`, `getline()`, `ignore()`, `ignore()`, `ignore()`, `putback()`, `read()`, `readsome()`, and `unget()`.

**`adjustfield`**

```
const fmtflags std::ios_base::adjustfield    [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**`app`**

```
const openmode std::ios_base::app    [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**`ate`**

```
const openmode std::ios_base::ate    [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

## badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char_type, traits_type >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< char_type, traits_type >::ws()`, and `std::ws()`.

## basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

## beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

## binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

## boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

## cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

## dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`.

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< char_type, traits_type >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::seekg()`, and `std::ws()`.

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< char_type, traits_type >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::seekp()`.

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by `std::fixed()`, and `std::hexfloat()`.

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_ostream< _CharT, _Traits >::flush()`.



`std::basic_ostream< char_type, traits_type >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::ws()` and `std::ws()`.

## hex

const `fmtflags` `std::ios_base::hex` [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::hex()`.

## in

const `openmode` `std::ios_base::in` [static], [inherited]

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::xsgetn()` and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

## internal

const `fmtflags` `std::ios_base::internal` [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by `std::internal()`.

## left

const `fmtflags` `std::ios_base::left` [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

## oct

const `fmtflags` `std::ios_base::oct` [static], [inherited]

Converts integer input or generates integer output in octal base.

Referenced by `std::oct()`.

## out

const `openmode` `std::ios_base::out` [static], [inherited]

Open for output. Default for `ofstream` and `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

## right

const `fmtflags` `std::ios_base::right` [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.) Referenced by [std::right\(\)](#).

### scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

### showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put< \\_CharT, \\_Outiter >::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

### showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

### showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

### skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), and [std::skipws\(\)](#).

### trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

### unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

### uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put< \\_CharT, \\_Outiter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [istream](#)

## 5.225 std::basic\_istream<\_CharT, \_Traits> Class Template Reference

```
#include <istream>
```

Inheritance diagram for std::basic\_istream<\_CharT, \_Traits>:



### Classes

- class [sentry](#)

### Public Types

- typedef [ctype](#)<\_CharT> **\_\_ctype\_type**
- typedef [basic\\_ios](#)<\_CharT, \_Traits> **\_\_ios\_type**
- typedef [basic\\_istream](#)<\_CharT, \_Traits> **\_\_istream\_type**
- typedef [num\\_get](#)<\_CharT, [istreambuf\\_iterator](#)<\_CharT, \_Traits>> **\_\_num\_get\_type**
- typedef [basic\\_streambuf](#)<\_CharT, \_Traits> **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_Traits::int\_type **int\_type**
- typedef \_Traits::int\_type **int\_type**
- typedef \_Traits::off\_type **off\_type**
- typedef \_Traits::pos\_type **pos\_type**
- typedef \_Traits::pos\_type **pos\_type**
- typedef \_Traits::seekdir **seekdir**
- typedef \_Traits **traits\_type**
- typedef [num\\_put](#)<\_CharT, [ostreambuf\\_iterator](#)<\_CharT, \_Traits>> **\_\_num\_put\_type**

**Public Member Functions**

- [basic\\_istream](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
  - virtual [~basic\\_istream](#) ()
  - template<typename [\\_ValueT](#)>  
[basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > & [M\\_extract](#) ([\\_ValueT](#) &\_\_v)
  - const [locale](#) & [M\\_getloc](#) () const
  - void [M\\_setstate](#) ([iostate](#) \_\_state)
  - bool [bad](#) () const
  - void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
  - [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
  - bool [eof](#) () const
  - [iostate](#) [exceptions](#) () const
  - void [exceptions](#) ([iostate](#) \_\_except)
  - bool [fail](#) () const
  - [char\\_type](#) [fill](#) () const
  - [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
  - [fmtflags](#) [flags](#) () const
  - [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
  - [streamsize](#) [gcount](#) () const
  - [basic\\_istream](#)< [char](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
  - [basic\\_istream](#)< [wchar\\_t](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
  - [locale](#) [getloc](#) () const
  - bool [good](#) () const
  - [basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n)
  - [basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n)
  - [basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
  - [basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
  - [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
  - long & [iword](#) (int \_\_ix)
  - [char](#) [narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (void \*&\_\_p)
  - [streamsize](#) [precision](#) () const
  - [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
  - void \*& [pword](#) (int \_\_ix)
  - [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) () const
  - [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
  - [iostate](#) [rdstate](#) () const
  - void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
  - void [setstate](#) ([iostate](#) \_\_state)
  - [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) () const
  - [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) ([basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_tiestr)
  - void [unsetf](#) ([fmtflags](#) \_\_mask)
  - [char\\_type](#) [widen](#) ([char](#) \_\_c) const
  - [streamsize](#) [width](#) () const
  - [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- 
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_istream\\_type](#) &(\_\_pf)(\_\_istream\_type &))

- `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type & __sb, char_type __delim)`
- `__istream_type & get (__streambuf_type & __sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator bool () const`
- `bool operator! () const`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool `__sync`=true)
- static int `xalloc` () throw ()

### Public Attributes

- class `__attribute__((__abi_tag__("cxx11")))` failure typedef `_ios_Fmtflags` `fmtflags`

### Static Public Attributes

- static const `openmode` `__noreplace`
- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- **basic\_istream** (**basic\_istream** &&\_\_rhs)
- **basic\_istream** (const **basic\_istream** &)=delete
- void **\_M\_cache\_locale** (const **locale** &\_\_loc)
- void **\_M\_call\_callbacks** (**event** \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_\_ValueT>  
  **\_istream\_type** & **\_M\_extract** (\_\_ValueT &\_\_v)
- **\_Words** & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- void **\_M\_move** (**ios\_base** &) noexcept
- void **\_M\_swap** (**ios\_base** &\_\_rhs) noexcept
- void **init** (**basic\_streambuf**< \_\_CharT, \_\_Traits > \*\_\_sb)
- void **move** (**basic\_ios** &&\_\_rhs)
- void **move** (**basic\_ios** &\_\_rhs)
- **basic\_istream** & **operator=** (**basic\_istream** &&\_\_rhs)
- **basic\_istream** & **operator=** (const **basic\_istream** &)=delete
- void **set\_rdbuf** (**basic\_streambuf**< \_\_CharT, \_\_Traits > \*\_\_sb)
- void **swap** (**basic\_ios** &\_\_rhs) noexcept
- void **swap** (**basic\_istream** &\_\_rhs)

## Protected Attributes

- **\_Callback\_list** \* **\_M\_callbacks**
- const **\_\_ctype\_type** \* **\_M\_ctype**
- **iostate** **\_M\_exception**
- **char\_type** **\_M\_fill**
- bool **\_M\_fill\_init**
- **fmtflags** **\_M\_flags**
- **streamsize** **\_M\_gcount**
- **locale** **\_M\_ios\_locale**
- **\_Words** **\_M\_local\_word** [**\_S\_local\_word\_size**]
- const **\_\_num\_get\_type** \* **\_M\_num\_get**
- const **\_\_num\_put\_type** \* **\_M\_num\_put**
- **streamsize** **\_M\_precision**
- **basic\_streambuf**< \_\_CharT, \_\_Traits > \* **\_M\_streambuf**
- **iostate** **\_M\_streambuf\_state**
- **basic\_ostream**< \_\_CharT, \_\_Traits > \* **\_M\_tie**
- **streamsize** **\_M\_width**
- **\_Words** \* **\_M\_word**
- int **\_M\_word\_size**
- **\_Words** **\_M\_word\_zero**

## Friends

- class **sentry**

### 5.225.1 Detailed Description

```
template<typename __CharT, typename __Traits>
class std::basic_istream< __CharT, __Traits >
```

Template class **basic\_istream**.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.

## 5.225.2 Member Typedef Documentation

`__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

`event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

`iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

`openmode`

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:



- app
- ate
- binary
- in
- out
- trunc

### **seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

## **5.225.3 Member Enumeration Documentation**

### **event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

## **5.225.4 Constructor & Destructor Documentation**

### **basic\_istream()**

```
template<typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits >::basic_istream (  
    __streambuf_type * __sb) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

### **~basic\_istream()**

```
template<typename _CharT, typename _Traits>  
virtual std::basic_istream< _CharT, _Traits >::~~basic_istream () [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

## **5.225.5 Member Function Documentation**

### **\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_year\(\)](#), [std::num\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::time\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), and [std::time\\_put< \\_CharT, \\_Outlter >::put\(\)](#).

**bad()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#).

**clear()**

```
template<typename _CharT, typename _Traits>
```

```
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::exceptions\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::unget\(\)](#), and [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#).

**copyfmt()**

```
template<typename _CharT, typename _Traits>
```

```
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by `std::basic_ios< char_type, traits_type >::operator bool()`, `std::basic_ios< char_type, traits_type >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

**fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
Retrieves the empty character.
```

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by `copyfmt()`, `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()()`.

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch) [inline], [inherited]
Sets a new empty character.
```

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
Access to format flags.
```

**Returns**

The format control flags for both input and output.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()()`, `std::chrono::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()()`.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline], [inherited]
Setting new format flags all at once.
```

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

References [fmtflags](#).

**gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
    void )
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb) [inline]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim)
```

Extraction into another streambuf.

## Parameters

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

## Returns

`*this`

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References [\\_M\\_gcount](#), and [std::ios\\_base::goodbit](#).

**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
    char_type & __c)
```

Simple extraction.

## Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

## Returns

`*this`

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

## Note

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::get (
    char_type * __s,
    streamsize __n) [inline]
```

Simple multiple-character extraction.

**Parameters**

|                                  |                                                           |
|----------------------------------|-----------------------------------------------------------|
| <a href="#"><code>__s</code></a> | Pointer to an array.                                      |
| <a href="#"><code>__n</code></a> | Maximum number of characters to store in <code>s</code> . |

**Returns**

`*this`

Returns `get(__s,__n,widen("\n"))`.

**get() [6/6]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n,
    char_type __delim)
```

Simple multiple-character extraction.

**Parameters**

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <a href="#"><code>__s</code></a>     | Pointer to an array.                                        |
| <a href="#"><code>__n</code></a>     | Maximum number of characters to store in <code>__s</code> . |
| <a href="#"><code>__delim</code></a> | A "stop" character.                                         |

**Returns**

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::](#)

**getline() [1/3]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n) [inline]
```

String extraction.

## Parameters

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

## Returns

\*this

Returns `getline(__s, __n, widen("\n"))`.

**getline()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim)
```

String extraction.

## Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

## Returns

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References `_M_gcount`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**getline()** [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim)
```

Explicit specialization declarations, defined in `src/istream.cc`.



**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::money\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

**good()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

**ignore()** [1/3]

```
template<typename _CharT, typename _Traits>
```

```
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (  
    void )
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#).

**ignore()** [2/3]

```
template<typename _CharT, typename _Traits>
```

```
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (  
    streamsize __n)
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

**ignore()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim)
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios<_CharT, _Traits>::imbue (
    const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::imbue\(\)](#), and [rdbuf\(\)](#).

Referenced by [std::chrono::operator<<\(\)](#).

**init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
    basic_streambuf<_CharT, _Traits> * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::basic\\_ios\(\)](#), and [std::basic\\_ostream<char\\_type, traits\\_type>::basic\\_ostream\(\)](#).

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline], [inherited]
```

Access to integer array.

## Parameters

|                         |                       |
|-------------------------|-----------------------|
| <code>_↔<br/>_ix</code> | Index into the array. |
|-------------------------|-----------------------|

## Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [iword\(\)](#).

Referenced by [iword\(\)](#).

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __c,
    char __dfault) const [inline], [inherited]
```

Squeezes characters.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

## Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator"!"()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    __ios_type &(* __pf ) (__ios_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type &(* __pf ) (__istream_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    __streambuf_type * __sb)
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_istream< _CharT, _Traits >::rdbuf()`, and `std::basic_istream< _CharT, _Traits >::setstate()`.

**operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>() [5/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    double & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>() [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    float & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>() [7/17]**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
    int & __n)
```

Integer arithmetic extractors.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::use_facet()`.

**operator>>() [8/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    ios_base &(* __pf ) (ios_base &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>() [9/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [10/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long double & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|                                                                                       |                                            |
|---------------------------------------------------------------------------------------|--------------------------------------------|
| $\leftrightarrow$<br>$\leftrightarrow$<br>$\leftrightarrow$<br>$\leftrightarrow$<br>f | A variable of builtin floating point type. |
|---------------------------------------------------------------------------------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
    short & __n)
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::use_facet()`.

**operator>>()** [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned int & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|



**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                                                                                                    |                                      |
|----------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>__n</code> | A variable of builtin integral type. |
|----------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                                                                                                      |                                      |
|------------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>__n</code> | A variable of builtin integral type. |
|------------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

\*this if successful


These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned short & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                                                                                                      |                                      |
|------------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>__n</code> | A variable of builtin integral type. |
|------------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    void *& __p) [inline]
```

Basic arithmetic extractors.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek (
    void )
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**putback()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c)
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sputbackc\(\)](#).

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Referenced by [clear\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [imbue\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellp\(\)](#), and [std::ws\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdstate()**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::bad\(\)](#), [clear\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::eof\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::fail\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::good\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#).

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
    char_type * __s,
    streamsize __n)
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
    char_type * __s,
    streamsize __n)
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called A here:

- if `A == -1`, sets `eofbit` and extracts no characters
- if `A == 0`, extracts no characters

- if  $A > 0$ , extracts  $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::](#) and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

### register\_callback()

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

#### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

References [fmtflags](#).

### seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    off_type __off,
    ios_base::seekdir __dir)
```

Changing the current read position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekoff(__off, __dir)`. If that function fails, sets `failbit`.

#### Note

This function first clears `eofbit`. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::](#) and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

### seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos)
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::failbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
    iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::getline()`, `std::getline()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, and `std::ws()`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream<_CharT, _Traits>::sync (
    void )
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>



**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void )
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References [std::ios\\_base::badbit](#), [std::ios\\_base::cur](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::in](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#).

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), and [copyfmt\(\)](#).

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
    void )
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References [fmtflags](#).

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::widen (
    char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::fill\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::subtract\\_with\\_carry\\_engine<uint\\_fast32\\_t, std::tr2::operator>>\(\)](#), and [std::subtract\\_with\\_carry\\_engine<uint\\_fast32\\_t, 24, 10, 24>::seed\(\)](#).

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::operator>>\(\)](#), and [std::operator>>\(\)](#).

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.225.6 Member Data Documentation****M\_gcount**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::_M_gcount [protected]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by [get\(\)](#), [get\(\)](#), [get\(\)](#), [get\(\)](#), [getline\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [putback\(\)](#), [read\(\)](#), [readsome\(\)](#), and [unget\(\)](#).

**adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::internal\(\)](#), [std::left\(\)](#), and [std::right\(\)](#).

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#).

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~sentry\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::clear\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_ostream< char\\_type, traits\\_type >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::tellg\(\)](#), and [std::ws\(\)](#).

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Referenced by [std::dec\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::hex\(\)](#), and [std::oct\(\)](#).

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekpos\(\)](#).

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#).

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract bool in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#).

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_time\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::getline\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#).

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_year\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::readsome\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#) and [std::ws\(\)](#).

## hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by [std::num\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), and [std::hex\(\)](#).

## in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Referenced by [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::showmanyc\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::underflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::xsgetn\(\)](#) and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#).

## internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

## left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by [std::num\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), and [std::left\(\)](#).

## oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

## out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#) and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::xsgetn\(\)](#).

`std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_stringbuf<_CharT, _Traits, std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

### right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Referenced by `std::right()`.

### scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by `std::hexfloat()`, and `std::scientific()`.

### showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::noshowbase()`, and `std::showbase()`.

### showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

### showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by `std::noshowpos()`, and `std::showpos()`.

### skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by `std::noskipws()`, `std::__detail::operator>>()`, and `std::skipws()`.

### trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

### unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::~~sentry()`, `std::nounitbuf()`, and `std::unitbuf()`.

### uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [istream](#)
- [istream.tcc](#)

## 5.226 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference

```
#include <sstream>
```

Inheritance diagram for std::basic\_istream< \_CharT, \_Traits, \_Alloc >:



### Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
- typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
- typedef [basic\\_istream](#)< char\_type, traits\_type > **\_\_istream\_type**
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**
- typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
- typedef [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **\_\_string\_type**
- typedef [basic\\_stringbuf](#)< \_CharT, \_Traits, \_Alloc > **\_\_stringbuf\_type**
- typedef \_Alloc **allocator\_type**
- typedef \_CharT **char\_type**
- enum [event](#) { **erase\_event** , **imbue\_event** , **copyfmt\_event** }
- typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
- typedef traits\_type::int\_type **int\_type**
- typedef \_ios\_ostate **iostate**
- typedef traits\_type::off\_type **off\_type**



- typedef `_ios_Openmode` `openmode`
  - typedef `traits_type::pos_type` `pos_type`
  - typedef `_ios_Seekdir` `seekdir`
  - typedef `_Traits` `traits_type`
- 
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`

## Public Member Functions

- `basic_istream` ()
- `basic_istream` (`__string_type` &&\_\_str, `ios_base::openmode` \_\_mode=`ios_base::in`)
- `basic_istream` (`basic_istream` &&\_\_rhs)
- `basic_istream` (const `__string_type` &\_\_str, `ios_base::openmode` \_\_mode=`ios_base::in`)
- `basic_istream` (const `basic_istream` &)=delete
- template<typename `_SAlloc`>  
  `basic_istream` (const `basic_string`< `_CharT`, `_Traits`, `_SAlloc` > &\_\_str, const `allocator_type` &\_\_a)
- template<typename `_SAlloc`>  
  `basic_istream` (const `basic_string`< `_CharT`, `_Traits`, `_SAlloc` > &\_\_str, `ios_base::openmode` \_\_mode, const `allocator_type` &\_\_a)
- template<typename `_SAlloc`>  
  `basic_istream` (const `basic_string`< `_CharT`, `_Traits`, `_SAlloc` > &\_\_str, `ios_base::openmode` \_\_mode=`ios_base::in`)
- `basic_istream` (`ios_base::openmode` \_\_mode)
- `basic_istream` (`ios_base::openmode` \_\_mode, const `allocator_type` &\_\_a)
- `~basic_istream` ()
- template<typename `_ValueT`>  
  `basic_istream`< `_CharT`, `_Traits` > & `M_extract` (`_ValueT` &\_\_v)
- const `locale` & `M_getloc` () const
- void `M_setstate` (`iostate` \_\_state)
- bool `bad` () const
- void `clear` (`iostate` \_\_state=`goodbit`)
- `basic_ios` & `copyfmt` (const `basic_ios` &\_\_rhs)
- bool `eof` () const
- `iostate exceptions` () const
- void `exceptions` (`iostate` \_\_except)
- bool `fail` () const
- `char_type fill` () const
- `char_type fill` (`char_type` \_\_ch)
- `fmtflags flags` () const
- `fmtflags flags` (`fmtflags` \_\_fmtfl)
- `streamsize gcount` () const
- `basic_istream`< `char` > & `getline` (`char_type` \*\_\_s, `streamsize` \_\_n, `char_type` \_\_delim)
- `basic_istream`< `wchar_t` > & `getline` (`char_type` \*\_\_s, `streamsize` \_\_n, `char_type` \_\_delim)
- `locale getloc` () const
- bool `good` () const
- `basic_istream`< `char` > & `ignore` (`streamsize` \_\_n)
- `basic_istream`< `wchar_t` > & `ignore` (`streamsize` \_\_n)
- `basic_istream`< `char` > & `ignore` (`streamsize` \_\_n, `int_type` \_\_delim)
- `basic_istream`< `wchar_t` > & `ignore` (`streamsize` \_\_n, `int_type` \_\_delim)
- `locale imbue` (const `locale` &\_\_loc)
- long & `word` (int \_\_ix)

- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- [basic\\_istream](#) & **operator=** ([basic\\_istream](#) &&\_\_rhs)
- [basic\\_istream](#) & **operator=** (const [basic\\_istream](#) &)=delete
- [\\_\\_istream\\_type](#) & **operator>>** ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- [\\_\\_istream\\_type](#) & **operator>>** (void \*&\_\_p)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- [\\_\\_stringbuf\\_type](#) \* [rdbuf](#) () const
- [iostate](#) [rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [setstate](#) ([iostate](#) \_\_state)
- [\\_\\_string\\_type](#) [str](#) () &&
- [\\_\\_string\\_type](#) [str](#) () const &
- void [str](#) ([\\_\\_string\\_type](#) &&\_\_s)
- void [str](#) (const [\\_\\_string\\_type](#) &\_\_s)
- template<\_\_allocator\_like \_SAlloc>  
[basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > [str](#) (const \_SAlloc &\_\_sa) const
- template<\_\_allocator\_like \_SAlloc>  
requires (is\_same\_v<\_SAlloc, \_Alloc>)  
void [str](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > &\_\_s)
- void [swap](#) ([basic\\_istream](#) &\_\_rhs)
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \*\_\_tiestr)
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- [basic\\_string\\_view](#)< char\_type, traits\_type > [view](#) () const noexcept
- char\_type [widen](#) (char \_\_c) const
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- [\\_\\_istream\\_type](#) & **operator>>** ([\\_\\_istream\\_type](#) &(\*\_\_pf)(\_\_istream\_type &))
- [\\_\\_istream\\_type](#) & **operator>>** ([\\_\\_ios\\_type](#) &(\*\_\_pf)(\_\_ios\_type &))
- [\\_\\_istream\\_type](#) & **operator>>** ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- [\\_\\_istream\\_type](#) & **operator>>** (bool &\_\_n)
- [\\_\\_istream\\_type](#) & **operator>>** (short &\_\_n)
- [\\_\\_istream\\_type](#) & **operator>>** (unsigned short &\_\_n)
- [\\_\\_istream\\_type](#) & **operator>>** (int &\_\_n)
- [\\_\\_istream\\_type](#) & **operator>>** (unsigned int &\_\_n)
- [\\_\\_istream\\_type](#) & **operator>>** (long &\_\_n)

- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`

- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type & __sb, char_type __delim)`
- `__istream_type & get (__streambuf_type & __sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`

- `operator bool () const`
- `bool operator! () const`

### Static Public Member Functions

- `static bool sync_with_stdio (bool __sync=true)`
- `static int xalloc () throw ()`

### Public Attributes

- `class __attribute__((__abi_tag__("cxx11"))) failure typedef _los_Fmtflags fmtflags`

### Static Public Attributes

- static const [openmode](#) `__noreplace`
- static const [fmtflags](#) `adjustfield`
- static const [openmode](#) `app`
- static const [openmode](#) `ate`
- static const [iostate](#) `badbit`
- static const [fmtflags](#) `basefield`
- static const [seekdir](#) `beg`
- static const [openmode](#) `binary`
- static const [fmtflags](#) `boolalpha`
- static const [seekdir](#) `cur`
- static const [fmtflags](#) `dec`
- static const [seekdir](#) `end`
- static const [iostate](#) `eofbit`
- static const [iostate](#) `failbit`
- static const [fmtflags](#) `fixed`
- static const [fmtflags](#) `floatfield`
- static const [iostate](#) `goodbit`
- static const [fmtflags](#) `hex`
- static const [openmode](#) `in`
- static const [fmtflags](#) `internal`
- static const [fmtflags](#) `left`
- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) & \_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- template<typename \_ValueT>  
[\\_\\_istream\\_type](#) & [\\_M\\_extract](#) (\_ValueT & \_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) & \_\_rhs) noexcept
- void [init](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
- void [move](#) ([basic\\_ios](#) && \_\_rhs)

- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) &\_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- char\_type **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)<\_CharT, \_Traits> \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

### 5.226.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_istream<_CharT, _Traits, _Alloc>
```

Controlling input for std::string.

#### Template Parameters

|                         |                                                                              |
|-------------------------|------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                    |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits</a> <_CharT>. |
| <a href="#">_Alloc</a>  | Allocator type, defaults to <a href="#">allocator</a> <_CharT>.              |

This class supports reading from objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as *sb*.

### 5.226.2 Member Typedef Documentation

#### [\\_\\_num\\_put\\_type](#)

```
template<typename _CharT, typename _Traits>
typedef num\_put<_CharT, ostreambuf\_iterator<_CharT, _Traits> > std::basic\_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

**event\_callback**

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

**Parameters**

|                        |                                                        |
|------------------------|--------------------------------------------------------|
| <code>_↔<br/>_e</code> | One of the members of the event enum.                  |
| <code>_↔<br/>_b</code> | Reference to the <code>ios_base</code> object.         |
| <code>_↔<br/>_i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

**iostate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

**openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

### 5.226.3 Member Enumeration Documentation

#### event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during imbue(). copyfmt\_event is used during copyfmt().

### 5.226.4 Constructor & Destructor Documentation

#### basic\_istream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream<_CharT, _Traits, _Alloc>::basic_istream () [inline]
```

Default constructor starts with an empty string buffer.

Initializes sb using in, and passes &sb to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

#### basic\_istream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream<_CharT, _Traits, _Alloc>::basic_istream (
    ios_base::openmode __mode) [inline], [explicit]
```

Starts with an empty string buffer.

#### Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

ios\_base::in is automatically included in `__mode`.

Initializes sb using `__mode|in`, and passes &sb to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

#### basic\_istream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream<_CharT, _Traits, _Alloc>::basic_istream (
    const __string_type & __str,
    ios_base::openmode __mode = ios_base::in) [inline], [explicit]
```

Starts with an existing string buffer.

#### Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

ios\_base::in is automatically included in `mode`.

Initializes sb using `str` and `mode|in`, and passes &sb to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

#### ~basic\_istream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream<_CharT, _Traits, _Alloc>::~~basic_istream () [inline]
```

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.



### 5.226.5 Member Function Documentation

#### **`_M_getloc()`**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::time_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

#### **`bad()`**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic\_ios<_CharT, _Traits>::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

##### Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

#### **`clear()`**

```
template<typename _CharT, typename _Traits>
```

```
void std::basic\_ios<_CharT, _Traits>::clear (  
    iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

##### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios<char_type, traits_type>::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream<_CharT, _Traits>::rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::unget()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

#### **`copyfmt()`**

```
template<typename _CharT, typename _Traits>
```

```
basic\_ios<_CharT, _Traits> & std::basic\_ios<_CharT, _Traits>::copyfmt (  
    const basic\_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

##### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof() const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::exceptions() const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::exceptions (
    iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
```

```
f.setstate (std::ios_base::badbit);

std::cerr << "Setting exception mask\n";
f.exceptions (std::ios_base::badbit);
}
```

### fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
Fast error checking.
```

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#), and [std::regex\\_traits< \\_Ch\\_type >::value\(\)](#).

### fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
Retrieves the empty character.
```

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Referenced by [copyfmt\(\)](#), [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#), and [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#).

### fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

### flags() [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t >::do\\_subtract\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator<<\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t >::do\\_subtract\(\)](#).

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

References [fmtflags](#).

**gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

`*this`

Returns `get(__sb,widen("\n"))`.

**get() [3/6]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

`*this`

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References [\\_M\\_gcount](#), and [std::ios\\_base::goodbit](#).

**get() [4/6]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c) [inherited]
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

`*this`

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                                                                                                                       |                                                           |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| <a href="#"></a><br><code>__s</code> | Pointer to an array.                                      |
| <a href="#"></a><br><code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

`*this`

Returns `get(__s,__n,widen("\n"))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

**getline()** [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                                                                                                                       |                                               |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| <a href="#"></a><br><code>__s</code> | A character array in which to store the data. |
| <a href="#"></a><br><code>__n</code> | Maximum number of characters to extract.      |

**Returns**

`*this`

Returns `getline(__s,__n,widen("\n"))`.

**getline()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case `eofbit` is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case `failbit` is set in the stream error state

If no characters are extracted, `failbit` is set. (An empty line of input should therefore not cause `failbit` to be set.)

In any case, a null character is stored in the next location in the array.

References [\\_M\\_gcount](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sgetc\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

**getline()** [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::operator<<()`, `std::operator>>()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::__detail::operator>>()`.

**ignore()** [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**ignore()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
    streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_streambuf<_CharT, _Traits>::rdbuf()`.



**ignore()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim) [inherited]
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::](#)

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::imbue\(\)](#), and [rdbuf\(\)](#).

Referenced by [std::chrono::operator<<\(\)](#).

**init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::basic\\_ios\(\)](#), and [std::basic\\_ostream< char\\_type, traits\\_type >::basic\\_ostream\(\)](#).

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [iword\(\)](#).

Referenced by [iword\(\)](#).

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __c,
    char __dfault) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator"!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator>>() [1/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    __ios_type &(* __pf )(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.

For more information, see the `io manip` header.

**operator>>() [2/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type &(* __pf )(__istream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.

For more information, see the `io manip` header.

**operator>>() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**operator>>() [4/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [5/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| $f$             |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| $f$             |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n) [inherited]
```

Integer arithmetic extractors.

## Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\leftarrow$   | A variable of builtin integral type. |
| $\leftarrow n$ |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>()** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    ios_base &(* __pf ) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\leftarrow$   | A variable of builtin integral type. |
| $\leftarrow n$ |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|                |                                            |
|----------------|--------------------------------------------|
| $\leftarrow$   | A variable of builtin floating point type. |
| $\leftarrow$   |                                            |
| $\leftarrow$   |                                            |
| $\leftarrow$   |                                            |
| $\leftarrow f$ |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

**operator>>()** [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| $n$          |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| $n$          |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| $n$          |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.



**operator>>()** [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**putback()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sputbackc\(\)](#).

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_istream< _CharT, _Traits, _Alloc >::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

**rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios< char_type, traits_type >::bad()`, `clear()`, `std::basic_ios< char_type, traits_type >::eof()`, `std::basic_ios< char_type, traits_type >::fail()`, `std::basic_ios< char_type, traits_type >::good()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::unget()`.

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
    char_type * __s,
    streamsize __n) [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__s</code> | A character array. |
|------------------|--------------------|

|                  |                                        |
|------------------|----------------------------------------|
| <code>__n</code> | Maximum number of characters to store. |
|------------------|----------------------------------------|

**Returns**

`*this`

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::readsome (
    char_type * __s,
    streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the `streambuf`'s buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets `eofbit` and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the `streambuf`.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

References [fmtflags](#).

**seekg() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    off_type __off,
    ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#) and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**seekg() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

## Note

This function first clears `eofbit`. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::failbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

## Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

## Parameters

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

## Returns

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
    iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::read\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), and [std::ws\(\)](#).

**str()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_istreamstream<_CharT, _Traits, _Alloc>::str () const & [inline],
[nodiscard]
```

Copying out the string buffer.

**Returns**

`rdbuf() -> str()`

**str()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_istreamstream<_CharT, _Traits, _Alloc>::str (
    const __string_type & __s) [inline]
```

Setting a new buffer.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Calls `rdbuf() -> str(s)`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream<_CharT, _Traits>::sync (
    void ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::pubsync\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void ) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf()->pubseekoff(0, cur, in).

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount(). At variance with putback, unget and seekg, eofbit is not cleared first.

References [std::ios\\_base::badbit](#), [std::ios\\_base::cur](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::in](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#).

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [copyfmt\(\)](#).

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.



**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**ungetc()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ungetc (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::rdstate()`.

Referenced by `std::__detail::operator>>()`.

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References `fmtflags`.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::basic_ios<char_type, traits_type>::fill()`, `std::getline()`, `std::getline()`, `std::subtract_with_carry_engine<uint_fast32_t, std::tr::operator>>()`, and `std::subtract_with_carry_engine<uint_fast32_t, 24, 10, 24>::seed()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outiter>::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.226.6 Member Data Documentation

### **`_M_gcount`**

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by `get()`, `get()`, `get()`, `get()`, `getline()`, `ignore()`, `ignore()`, `ignore()`, `putback()`, `read()`, `readsome()`, and `unget()`.

### **`adjustfield`**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

### **`app`**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

### **`ate`**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

### **`badbit`**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char_type, traits_type >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::clear()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_ostream< char_type, traits_type >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< char_type, traits_type >::ws()`, and `std::ws()`.

### **`basefield`**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

### **`beg`**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary>.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#).

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract bool in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#).

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_time\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::getline\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), and [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_weekday\(\)](#).

`std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<char_type, traits_type>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, and `std::basic_ostream<_CharT, _Traits>::seekp()`.

## fixed

const `fmtflags` `std::ios_base::fixed` [static], [inherited]

Generate floating-point output in fixed-point notation.

Referenced by `std::fixed()`, and `std::hexfloat()`.

## floatfield

const `fmtflags` `std::ios_base::floatfield` [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

## goodbit

const `iosstate` `std::ios_base::goodbit` [static], [inherited]

Indicates all is well.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<char_type, traits_type>::seekg()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::ws()`, and `std::ws()`.

## hex

const `fmtflags` `std::ios_base::hex` [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::hex()`.

## in

const `openmode` `std::ios_base::in` [static], [inherited]

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

## internal

const `fmtflags` `std::ios_base::internal` [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

### **left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by [std::num\\_put<\\_CharT, \\_Outiter>::do\\_put\(\)](#), and [std::left\(\)](#).

### **oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

### **out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits, \\_Alloc>::xsputn\(\)](#).

### **right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Referenced by [std::right\(\)](#).

### **scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

### **showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put<\\_CharT, \\_Outiter>::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

### **showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

### **showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

const [fmtflags](#) std::ios\_base::skipws [static], [inherited]

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), and [std::skipws\(\)](#).

**trunc**

const [openmode](#) std::ios\_base::trunc [static], [inherited]

Truncate an existing stream when opening. Default for ofstream.

**unitbuf**

const [fmtflags](#) std::ios\_base::unitbuf [static], [inherited]

Flushes output after each output operation.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

const [fmtflags](#) std::ios\_base::uppercase [static], [inherited]

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)

**5.227 std::basic\_ofstream<\_CharT, \_Traits > Class Template Reference**

```
#include <fstream>
```

Inheritance diagram for std::basic\_ofstream< \_CharT, \_Traits >:



### Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
  - typedef [basic\\_filebuf](#)< char\_type, traits\_type > **\_\_filebuf\_type**
  - typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
  - typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**
  - typedef [basic\\_ostream](#)< char\_type, traits\_type > **\_\_ostream\_type**
  - typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
  - typedef \_CharT **char\_type**
  - enum [event](#) { [erase\\_event](#) , [imbue\\_event](#) , [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
  - typedef traits\_type::int\_type **int\_type**
  - typedef \_ios\_istate [iostate](#)
  - typedef traits\_type::off\_type **off\_type**
  - typedef \_ios\_Openmode [openmode](#)
  - typedef traits\_type::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir [seekdir](#)
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**



## Public Member Functions

- [basic\\_ofstream](#) ()
- **basic\_ofstream** ([basic\\_ofstream](#) && \_\_rhs)
- template<typename \_Path, typename \_Require = \_If\_fs\_path<\_Path>>  
  [basic\\_ofstream](#) (const \_Path & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- **basic\_ofstream** (const [basic\\_ofstream](#) &)=delete
- [basic\\_ofstream](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ofstream](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [~basic\\_ofstream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename \_ValueT>  
  [basic\\_ostream](#)<\_CharT, \_Traits> & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags flags](#) () const
- [fmtflags flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [locale getloc](#) () const
- bool [good](#) () const
- [locale imbue](#) (const [locale](#) & \_\_loc)
- bool [is\\_open](#) ()
- bool [is\\_open](#) () const
- long & [iword](#) (int \_\_ix)
- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- template<typename \_Path>  
  \_If\_fs\_path<\_Path, void> [open](#) (const \_Path & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- void [open](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- void [open](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_streambuf\\_type](#) \* \_\_sb)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (const void \* \_\_p)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (nullptr\_t)
- [basic\\_ofstream](#) & [operator=](#) ([basic\\_ofstream](#) && \_\_rhs)
- [basic\\_ofstream](#) & [operator=](#) (const [basic\\_ofstream](#) &)=delete
- [streamsize precision](#) () const
- [streamsize precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* [rdbuf](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- [\\_\\_filebuf\\_type](#) \* [rdbuf](#) () const
- [iostate rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [\\_\\_ostream\\_type](#) & [seekp](#) (off\_type, [ios\\_base::seekdir](#))

- `__ostream_type` & `seekp` (`pos_type`)
- `fmtflags` `setf` (`fmtflags` `__fmtfl`)
- `fmtflags` `setf` (`fmtflags` `__fmtfl`, `fmtflags` `__mask`)
- void `setstate` (`iosstate` `__state`)
- void `swap` (`basic_ofstream` & `__rhs`)
- `pos_type` `tellp` ()
- `basic_ofstream`< `_CharT`, `_Traits` > \* `tie` () const
- `basic_ofstream`< `_CharT`, `_Traits` > \* `tie` (`basic_ofstream`< `_CharT`, `_Traits` > \* `__tiestr`)
- void `unsetf` (`fmtflags` `__mask`)
- `char_type` `widen` (`char` `__c`) const
- `streamsize` `width` () const
- `streamsize` `width` (`streamsize` `__wide`)
- `__ostream_type` & `operator<<` ( `__ostream_type` & (\*`__pf`)(`__ostream_type` &))
- `__ostream_type` & `operator<<` ( `__ios_type` & (\*`__pf`)(`__ios_type` &))
- `__ostream_type` & `operator<<` (`ios_base` & (\*`__pf`)(`ios_base` &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ofstream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type` & `operator<<` (`long` `__n`)
- `__ostream_type` & `operator<<` (`unsigned long` `__n`)
- `__ostream_type` & `operator<<` (`bool` `__n`)
- `__ostream_type` & `operator<<` (`short` `__n`)
- `__ostream_type` & `operator<<` (`unsigned short` `__n`)
- `__ostream_type` & `operator<<` (`int` `__n`)
- `__ostream_type` & `operator<<` (`unsigned int` `__n`)
- `__ostream_type` & `operator<<` (`long long` `__n`)
- `__ostream_type` & `operator<<` (`unsigned long long` `__n`)
- `__ostream_type` & `operator<<` (`double` `__f`)
- `__ostream_type` & `operator<<` (`float` `__f`)
- `__ostream_type` & `operator<<` (`long double` `__f`)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ofstream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type` & `put` (`char_type` `__c`)
- `__ostream_type` & `write` (`const char_type` \*`__s`, `streamsize` `__n`)
- `operator bool` () const
- `bool operator!` () const

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Public Attributes

- class \_\_attribute\_\_((\_\_abi\_tag\_\_("cxx11"))) failure typedef \_ios\_Fmtflags [fmtflags](#)

### Static Public Attributes

- static const [openmode](#) \_\_noreplace
- static const [fmtflags](#) adjustfield
- static const [openmode](#) app
- static const [openmode](#) ate
- static const [iosstate](#) badbit
- static const [fmtflags](#) basefield
- static const [seekdir](#) beg
- static const [openmode](#) binary
- static const [fmtflags](#) boolalpha
- static const [seekdir](#) cur
- static const [fmtflags](#) dec
- static const [seekdir](#) end
- static const [iosstate](#) eofbit
- static const [iosstate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iosstate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

**Protected Member Functions**

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT>  
  [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_ostream](#) &\_\_rhs)

**Protected Attributes**

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)<\_CharT, \_Traits> \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

**5.227.1 Detailed Description**

```
template<typename _CharT, typename _Traits>
```

```
class std::basic_ofstream<_CharT, _Traits>
```

Controlling output for files.

**Template Parameters**

|                         |                                                                              |
|-------------------------|------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                    |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits</a> <_CharT>. |

This class supports reading from named files, using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as [sb](#).

## 5.227.2 Member Typedef Documentation

### `__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]
```

These are non-standard types.

### `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the ios_base object.                      |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

### `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

### `openmode`

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

**5.227.3 Member Enumeration Documentation****event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

**5.227.4 Constructor & Destructor Documentation****basic\_ofstream() [1/4]**

```
template<typename _CharT, typename _Traits>
```

```
std::basic_ofstream< _CharT, _Traits >::basic_ofstream () [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

**basic\_ofstream() [2/4]**

```
template<typename _CharT, typename _Traits>
```

```
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
    const char * __s,
    ios_base::openmode __mode = ios_base::out) [inline], [explicit]
```

Create an output file stream.

**Parameters**

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::out` is automatically included in `__mode`.

**basic\_ofstream() [3/4]**

```
template<typename _CharT, typename _Traits>
```

```
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::out) [inline], [explicit]
```

Create an output file stream.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__s</code> | <code>std::string</code> specifying the filename. |
|------------------|---------------------------------------------------|

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |
|---------------------|----------------------------------------------------------------|

`ios_base::out` is automatically included in `__mode`.

### **basic\_ofstream()** [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
    const _Path & __s,
    ios_base::openmode __mode = ios_base::out) [inline]
```

Create an output file stream.

#### Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | filesystem::path specifying the filename.                      |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::out` is automatically included in `__mode`.

### **~basic\_ofstream()**

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::~~basic_ofstream () [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

## 5.227.5 Member Function Documentation

### **\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_get< _CharT, _Inlter >::do_get_year()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

### **bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::clear (
    iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios<char_type, traits_type>::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream<_CharT, _Traits>::rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::unget()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**close()**

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream<_CharT, _Traits>::close () [inline]
```

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
    const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References `basic_ios()`, `std::__addressof()`, `exceptions()`, `fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `tie()`, `std::tie()`, and `std::ios_base::width()`.

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.



**exceptions()** [1/2]

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
Throwing exceptions on errors.
```

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by `clear()`, and `copyfmt()`.

**exceptions()** [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iosstate __except) [inline], [inherited]
Throwing exceptions on errors.
```

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
Fast error checking.
```

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Referenced by `std::basic_ios< char_type, traits_type >::operator bool()`, `std::basic_ios< char_type, traits_type >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellg()`, and `std::regex_traits< _Ch_type >::value()`.



**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush () [inherited]
Synchronizing the stream buffer.
```

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits`

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
Locale access.
```

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, `std::basic_ios< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard], [inherited]
Fast error checking.
```

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::__detail::operator>>()`.

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References `std::ios_base::getloc()`, `std::ios_base::imbue()`, and `rdbuf()`.

Referenced by `std::chrono::operator<<()`.

**init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
    basic_streambuf<_CharT, _Traits> * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_ios<char_type, traits_type>::basic_ios()`, and `std::basic_ostream<char_type, traits_type>::basic_ostream()`.

**is\_open()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ofstream<_CharT, _Traits>::is_open () [inline], [nodiscard]
```

Wrapper to test for an open file.

**Returns**

`rdbuf()->is_open()`

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References `iword()`.

Referenced by `iword()`.

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
```

```
char_type __c,  
char __dfault) const [inline], [inherited]
```

Squeezes characters.

## Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

## Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**open()** [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, void > std::basic_ofstream< _CharT, _Traits >::open (
    const _Path & __s,
    ios_base::openmode __mode = ios_base::out) [inline]
```

Opens an external file.

## Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The name of the file, as a filesystem::path. |
| <code>__mode</code> | The open mode flags.                         |

Calls `std::basic_filebuf::open(__s,__mode|out)`. If that function fails, `failbit` is set in the stream's error state.

**open()** [2/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::open (
    const char * __s,
    ios_base::openmode __mode = ios_base::out) [inline]
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s,__mode|out)`. If that function fails, `failbit` is set in the stream's error state.

**open()** [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::open (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::out) [inline]
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(s,mode|out)`. If that function fails, `failbit` is set in the stream's error state.

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator"!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(* __pf )(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

**operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(* __pf )(__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

**operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

#### operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
    bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

##### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

##### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
    const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

##### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

##### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
    double __f) [inline], [inherited]
```

Floating point arithmetic inserters.



**Parameters**

|                                                                                   |                                            |
|-----------------------------------------------------------------------------------|--------------------------------------------|
|  | A variable of builtin floating point type. |
|  |                                            |
|  |                                            |
|  |                                            |
| <i>f</i>                                                                          |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [7/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                                                                                     |                                            |
|-------------------------------------------------------------------------------------|--------------------------------------------|
|    | A variable of builtin floating point type. |
|    |                                            |
|    |                                            |
|  |                                            |
| <i>f</i>                                                                            |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                     |                                      |
|-------------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <i>n</i>                                                                            |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream<_CharT, _Traits>::operator<< (
    ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<()** [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream<_CharT, _Traits>::operator<< (
    long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream<_CharT, _Traits>::operator<< (
    long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream<_CharT, _Traits>::operator<< (
    long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    short __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put< \\_CharT, \\_Outiter >::put\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<()** [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
    unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
    unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits](#)

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_ofstream< _CharT, _Traits >::rdbuf () const [inline], [nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

**rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios< char_type, traits_type >::bad()`, `clear()`, `std::basic_ios< char_type, traits_type >::eof()`,

`std::basic_ios< char_type, traits_type >::fail()`, `std::basic_ios< char_type, traits_type >::good()`, `std::basic_istream< _CharT, _Traits >::`

`std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type >::`

and `std::basic_istream< _CharT, _Traits >::unget()`.

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered. References [fmtflags](#).

**seekp()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    off_type __off,
    ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

## Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

## Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**seekp()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    pos_type __pos) [inherited]
```

Changing the current write position.

## Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

## Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

References [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

## Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

## Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

## Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), and [std::ws\(\)](#).



**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (  
    bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tell()**

```
template<typename _CharT, typename _Traits>
basic_ofstream<_CharT, _Traits>::pos_type std::basic_ofstream<_CharT, _Traits>::tellp () [inherited]
Getting the current write position.
```

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

References `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`

**tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ofstream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
Fetches the current tied stream.
```

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ofstream<_CharT, _Traits>::sentry::sentry()`, and `copyfmt()`.

**tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ofstream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie (
    basic_ofstream<_CharT, _Traits> * __tiestr) [inline], [inherited]
Ties this stream to an output stream.
```

## Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References [fmtflags](#).

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::fill\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, std::tr2::operator>>\(\)](#), and [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::seed\(\)](#).

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::num\\_put< \\_CharT, \\_Outiter >::do\\_put\(\)](#), [std::operator>>\(\)](#), and [std::operator>>\(\)](#).

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

## Returns

The previous value of width().

**write()**

```
template<typename _CharT, typename _Traits>
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n) [inherited]
```

Character string insertion.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

## Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

## Note

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static], [inherited]
```

Access to unique indices.

## Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.227.6 Member Data Documentation

#### adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put<_CharT, _Outlter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

#### ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

#### badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<char_type, traits_type>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ostream<_CharT, _Traits>::clear()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_ostream<char_type, traits_type>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<char_type, traits_type>::seekg()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_istream<char_type, traits_type>::tellg()`, and `std::ws()`.

#### basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::num_put<_CharT, _Outlter>::do_put()`, `std::hex()`, and `std::oct()`.

#### beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf<_CharT, _Traits>::seekpos()`.

#### binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#).

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_time\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::getline\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#).

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::readsome\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), and [std::ws\(\)](#).

**hex**

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::hex\(\)](#).

**in**

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Referenced by [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::showmanyc\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::underflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::xsgetn\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#).

**internal**

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

**left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.) Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

### **oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by `std::oct()`.

### **out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekp()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

### **right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Referenced by `std::right()`.

### **scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by `std::hexfloat()`, and `std::scientific()`.

### **showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::noshowbase()`, and `std::showbase()`.

### **showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

### **showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by `std::noshowpos()`, and `std::showpos()`.

### **skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by `std::noskipws()`, `std::__detail::operator>>()`, and `std::skipws()`.



**trunc**

const [openmode](#) std::ios\_base::trunc [static], [inherited]  
 Truncate an existing stream when opening. Default for ofstream.

**unitbuf**

const [fmtflags](#) std::ios\_base::unitbuf [static], [inherited]  
 Flushes output after each output operation.  
 Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

const [fmtflags](#) std::ios\_base::uppercase [static], [inherited]  
 Replaces certain lowercase letters with their uppercase equivalents in generated output.  
 Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).  
 The documentation for this class was generated from the following file:

- [fstream](#)

**5.228 std::basic\_ostream< \_CharT, \_Traits > Class Template Reference**

#include <ostream.h>

Inheritance diagram for std::basic\_ostream< \_CharT, \_Traits >:

**Classes**

- class [sentry](#)

**Public Types**

- typedef [ctype< \\_CharT > \\_\\_ctype\\_type](#)

- typedef [basic\\_ios](#)<\_CharT, \_Traits> [\\_\\_ios\\_type](#)
  - typedef [num\\_put](#)<\_CharT, [ostreambuf\\_iterator](#)<\_CharT, \_Traits>> [\\_\\_num\\_put\\_type](#)
  - typedef [basic\\_ostream](#)<\_CharT, \_Traits> [\\_\\_ostream\\_type](#)
  - typedef [basic\\_streambuf](#)<\_CharT, \_Traits> [\\_\\_streambuf\\_type](#)
  - typedef \_CharT [char\\_type](#)
  - enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
  - typedef \_Traits::int\_type [int\\_type](#)
  - typedef \_ios::iostate [iostate](#)
  - typedef \_Traits::off\_type [off\\_type](#)
  - typedef \_ios::Openmode [openmode](#)
  - typedef \_Traits::pos\_type [pos\\_type](#)
  - typedef \_ios::Seekdir [seekdir](#)
  - typedef \_Traits [traits\\_type](#)
- 
- typedef [num\\_get](#)<\_CharT, [istreambuf\\_iterator](#)<\_CharT, \_Traits>> [\\_\\_num\\_get\\_type](#)

### Public Member Functions

- [basic\\_ostream](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- virtual [~basic\\_ostream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename \_ValueT>  
  [basic\\_ostream](#)<\_CharT, \_Traits> & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)
- [char](#) [narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (const void \*\_\_p)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (nullptr\_t)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* [rdbuf](#) () const
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* [rdbuf](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- [iostate](#) [rdstate](#) () const

- void `register_callback` (`event_callback` \_\_fn, int \_\_index)
- `__ostream_type` & `seekp` (`off_type`, `ios_base::seekdir`)
- `__ostream_type` & `seekp` (`pos_type`)
- `fmtflags` `setf` (`fmtflags` \_\_fmtfl)
- `fmtflags` `setf` (`fmtflags` \_\_fmtfl, `fmtflags` \_\_mask)
- void `setstate` (`iostate` \_\_state)
- `pos_type` `tellp` ()
- `basic_ostream`< `_CharT`, `_Traits` > \* `tie` () const
- `basic_ostream`< `_CharT`, `_Traits` > \* `tie` (`basic_ostream`< `_CharT`, `_Traits` > \* \_\_tiestr)
- void `unsetf` (`fmtflags` \_\_mask)
- `char_type` `widen` (`char` \_\_c) const
- `streamsize` `width` () const
- `streamsize` `width` (`streamsize` \_\_wide)
- `__ostream_type` & `operator<<` (`__ostream_type` &(\*\_\_pf)(\_\_ostream\_type &))
- `__ostream_type` & `operator<<` (`__ios_type` &(\*\_\_pf)(\_\_ios\_type &))
- `__ostream_type` & `operator<<` (`ios_base` &(\*\_\_pf)(ios\_base &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type` & `operator<<` (long \_\_n)
- `__ostream_type` & `operator<<` (unsigned long \_\_n)
- `__ostream_type` & `operator<<` (bool \_\_n)
- `__ostream_type` & `operator<<` (short \_\_n)
- `__ostream_type` & `operator<<` (unsigned short \_\_n)
- `__ostream_type` & `operator<<` (int \_\_n)
- `__ostream_type` & `operator<<` (unsigned int \_\_n)
- `__ostream_type` & `operator<<` (long long \_\_n)
- `__ostream_type` & `operator<<` (unsigned long long \_\_n)
- `__ostream_type` & `operator<<` (double \_\_f)
- `__ostream_type` & `operator<<` (float \_\_f)
- `__ostream_type` & `operator<<` (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type` & `put` (`char_type` \_\_c)
- `__ostream_type` & `write` (const `char_type` \*\_\_s, `streamsize` \_\_n)
- `operator bool` () const
- bool `operator!` () const

### Static Public Member Functions

- static bool `sync_with_stdio` (bool `__sync`=true)
- static int `xalloc` () throw ()

### Public Attributes

- class `__attribute__((__abi_tag__("cxx11")))` failure typedef `_ios_Fmtflags` `fmtflags`

### Static Public Attributes

- static const `openmode` `__noreplace`
- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- **basic\_ostream** ([basic\\_ostream](#)< \_CharT, \_Traits > &)
- **basic\_ostream** ([basic\\_ostream](#) &&\_\_rhs)
- **basic\_ostream** (const [basic\\_ostream](#) &)=delete
- void **\_M\_cache\_locale** (const [locale](#) & \_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- \_Words & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT>  
  [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- [basic\\_ostream](#) & **operator=** ([basic\\_ostream](#) &&\_\_rhs)
- [basic\\_ostream](#) & **operator=** (const [basic\\_ostream](#) &)=delete
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_ostream](#) &\_\_rhs)

## Protected Attributes

- \_Callback\_list \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- char\_type **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [locale](#) **\_M\_ios\_locale**
- \_Words **\_M\_local\_word** [\_S\_local\_word\_size]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- \_Words \* **\_M\_word**
- int **\_M\_word\_size**
- \_Words **\_M\_word\_zero**

## Friends

- class **sentry**

### 5.228.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ostream< _CharT, _Traits >
```

Template class basic\_ostream.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.

## 5.228.2 Member Typedef Documentation

`__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]
```

These are non-standard types.

`event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

`iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

`openmode`

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

### seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

## 5.228.3 Member Enumeration Documentation

### event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

## 5.228.4 Constructor & Destructor Documentation

### basic\_ostream()

```
template<typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits >::basic_ostream (  
    __streambuf_type * __sb) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Referenced by `operator<<()`.

### ~basic\_ostream()

```
template<typename _CharT, typename _Traits>  
virtual std::basic_ostream< _CharT, _Traits >::~~basic_ostream () [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

## 5.228.5 Member Function Documentation

### \_M\_getloc()

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get_date()`, `std::time_get<_CharT, _Inlter>::do_get_monthname()`, `std::time_get<_CharT, _Inlter>::do_get_weekday()`, `std::time_get<_CharT, _Inlter>::do_get_year()`, `std::num_put<_CharT, _Outlter>::do_put()`, `std::time_put<_CharT, _Outlter>::do_put()`, and `std::time_put<_CharT, _Outlter>::put()`.

**bad()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios<_CharT, _Traits>::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

**clear()**

```
template<typename _CharT, typename _Traits>
```

```
void std::basic_ios<_CharT, _Traits>::clear (
    iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios<char_type, traits_type>::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream<_CharT, _Traits>::rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::unget()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
```

```
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
    const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References `basic_ios()`, `std::__addressof()`, `exceptions()`, `fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `tie()`, `std::tie()`, and `std::ios_base::width()`.



**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by `std::basic_ios< char_type, traits_type >::operator bool()`, `std::basic_ios< char_type, traits_type >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

**fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by `copyfmt()`, `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()()`.

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()()`, `std::chrono::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()()`.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

References [fmtflags](#).

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()
Synchronizing the stream buffer.
```

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits](#)

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
Locale access.
```

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::money\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::do\\_get\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard], [inherited]
Fast error checking.
```

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc) [inherited]
Moves to a new locale.
```

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References `std::ios_base::getloc()`, `std::ios_base::imbue()`, and `rdbuf()`.

Referenced by `std::chrono::operator<<()`.

**init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
    basic_streambuf<_CharT, _Traits> * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_ios<char_type, traits_type>::basic_ios()`, and `std::basic_ostream<char_type, traits_type>::basic_ostream()`.

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References `iword()`.

Referenced by `iword()`.

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __C,
    char __dfault) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(* __pf )(__ios_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

**operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(* __pf )(__ostream_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

**operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb)
```

Extracting from another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

**operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    bool __n) [inline]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    const void * __p) [inline]
```

Pointer arithmetic inserters.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    double __f) [inline]
```

Floating point arithmetic inserters.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| <code>↔</code>  | A variable of builtin floating point type. |
| <code>_↔</code> |                                            |
| <code>↔</code>  |                                            |
| <code>_↔</code> |                                            |
| <code>f</code>  |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**`operator<<()` [7/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    float __f) [inline]
```

Floating point arithmetic inserters.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| <code>↔</code>  | A variable of builtin floating point type. |
| <code>_↔</code> |                                            |
| <code>↔</code>  |                                            |
| <code>_↔</code> |                                            |
| <code>f</code>  |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**`operator<<()` [8/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    int __n)
```

Integer arithmetic inserters.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.



**operator<<()** [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(* __pf ) (ios_base &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<()** [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    long double __f) [inline]
```

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    long long __n) [inline]
```

Integer arithmetic inserters.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    short __n)
```

Integer arithmetic inserters.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator<\\_CharT, \\_Traits>::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put<\\_CharT, \\_Outiter>::put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<()** [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned int __n) [inline]
```

Integer arithmetic inserters.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned long __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>_↔<br/>_n</code> | A variable of builtin integral type. |
|------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [16/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>_↔<br/>_n</code> | A variable of builtin integral type. |
|------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [17/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned short __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>_↔<br/>_n</code> | A variable of builtin integral type. |
|------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**precision() [1/2]**

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), and [std::chrono::operator<<\(\)](#).

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of precision().

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (
    char_type __c)
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>](#)

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Referenced by [clear\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::getline\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [imbue\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellp\(\)](#), and [std::ws\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdstate()**

```
template<typename _CharT, typename _Traits>
ios_state std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios<char_type, traits_type>::bad()`, `clear()`, `std::basic_ios<char_type, traits_type>::eof()`,

`std::basic_ios<char_type, traits_type>::fail()`, `std::basic_ios<char_type, traits_type>::good()`, `std::basic_istream<_CharT, _Traits>::`

`std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::`

and `std::basic_istream<_CharT, _Traits>::unget()`.

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

References [fmtflags](#).

**seekp() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
    off_type __off,
    ios_base::seekdir __dir)
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`,

and `std::basic_ios<_CharT, _Traits>::setstate()`.

**seekp() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
    pos_type __pos)
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets `failbit`.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf() [1/2]**

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf() [2/2]**

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
    iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

## Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::getline()`, `std::getline()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, and `std::ws()`.

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp ()
```

Getting the current write position.

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

References `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `copyfmt()`.



**tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References [fmtflags](#).

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
| <code>__c</code> |                         |

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::basic_ios< char_type, traits_type >::fill()`, `std::getline()`, `std::getline()`, `std::subtract_with_carry_engine< uint_fast32_t, std::tr2::operator>>()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::seed()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outiter>::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::write (
    const char_type * __s,
    streamsize __n)
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

`*this`

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.228.6 Member Data Documentation****adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char_type, traits_type >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::clear()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_ostream< char_type, traits_type >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::sync()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< char_type, traits_type >::ws()`, and `std::ws()`.

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

**beg**

const `seekdir` std::ios\_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekpos\(\)](#).

**binary**

const `openmode` std::ios\_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary>.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::showmanyc\(\)](#).

**boolalpha**

const `fmtflags` std::ios\_base::boolalpha [static], [inherited]

Insert/extract bool in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

**cur**

const `seekdir` std::ios\_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::imbue\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), and [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#).

**dec**

const `fmtflags` std::ios\_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#).

**end**

const `seekdir` std::ios\_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::open\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#).

**eofbit**

const `istate` std::ios\_base::eofbit [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_date\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_time\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::read\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::read\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::seekg\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_weekday\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_year\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::read\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), and [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#).

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_year\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::readsome\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ws\(\)](#), and [std::ws\(\)](#).

**hex**

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by [std::num\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), and [std::hex\(\)](#).

**in**

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Referenced by [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#).

`std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

### internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by `std::internal()`.

### left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, and `std::left()`.

### oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by `std::oct()`.

### out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekp()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

### right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Referenced by `std::right()`.

### scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by `std::hexfloat()`, and `std::scientific()`.

### showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::noshowbase()`, and `std::showbase()`.

### showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), and [std::skipws\(\)](#).

**trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

**unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [ostream.h](#)
- [ostream.tcc](#)

**5.229 std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference**

```
#include <sstream>
```

Inheritance diagram for std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >:



## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
  - typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
  - typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**
  - typedef [basic\\_ostream](#)< [char\\_type](#), [traits\\_type](#) > **\_\_ostream\_type**
  - typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
  - typedef [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **\_\_string\_type**
  - typedef [basic\\_stringbuf](#)< \_CharT, \_Traits, \_Alloc > **\_\_stringbuf\_type**
  - typedef \_Alloc **allocator\_type**
  - typedef \_CharT **char\_type**
  - enum [event](#) { [erase\\_event](#) , [imbue\\_event](#) , [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
  - typedef [traits\\_type](#)::int\_type **int\_type**
  - typedef \_ios\_istate [iostate](#)
  - typedef [traits\\_type](#)::off\_type **off\_type**
  - typedef \_ios\_Openmode [openmode](#)
  - typedef [traits\\_type](#)::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir [seekdir](#)
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**



## Public Member Functions

- [basic\\_ostringstream](#) ()
- [basic\\_ostringstream](#) ([\\_\\_string\\_type](#) && \_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ostringstream](#) ([basic\\_ostringstream](#) && \_\_rhs)
- [basic\\_ostringstream](#) (const [\\_\\_string\\_type](#) & \_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ostringstream](#) (const [basic\\_ostringstream](#) &)=delete
- [template<typename \\_SAlloc>](#)  
[basic\\_ostringstream](#) (const [basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_SAlloc](#) > & \_\_str, const [allocator\\_type](#) & \_\_a)
- [template<typename \\_SAlloc>](#)  
[basic\\_ostringstream](#) (const [basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_SAlloc](#) > & \_\_str, [ios\\_base::openmode](#) \_\_mode, const [allocator\\_type](#) & \_\_a)
- [template<typename \\_SAlloc>](#)  
[basic\\_ostringstream](#) (const [basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_SAlloc](#) > & \_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ostringstream](#) ([ios\\_base::openmode](#) \_\_mode)
- [basic\\_ostringstream](#) ([ios\\_base::openmode](#) \_\_mode, const [allocator\\_type](#) & \_\_a)
- [~basic\\_ostringstream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- [template<typename \\_ValueType>](#)  
[basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > & [\\_M\\_insert](#) ([\\_ValueType](#) \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) & \_\_loc)
- long & [iword](#) (int \_\_ix)
- [char](#) [narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_streambuf\\_type](#) \* \_\_sb)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (const void \* \_\_p)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (nullptr\_t)
- [basic\\_ostringstream](#) & [operator=](#) ([basic\\_ostringstream](#) && \_\_rhs)
- [basic\\_ostringstream](#) & [operator=](#) (const [basic\\_ostringstream](#) &)=delete
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- [\\_\\_stringbuf\\_type](#) \* [rdbuf](#) () const
- [iostate](#) [rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [\\_\\_ostream\\_type](#) & [seekp](#) ([off\\_type](#), [ios\\_base::seekdir](#))

- [\\_\\_ostream\\_type](#) & [seekp](#) (pos\_type)
  - [fmtflags](#) [setf](#) (fmtflags \_\_fmtfl)
  - [fmtflags](#) [setf](#) (fmtflags \_\_fmtfl, [fmtflags](#) \_\_mask)
  - void [setstate](#) (iostate \_\_state)
  - [\\_\\_string\\_type](#) [str](#) () &&
  - [\\_\\_string\\_type](#) [str](#) () const &
  - void [str](#) ([\\_\\_string\\_type](#) &&\_\_s)
  - void [str](#) (const [\\_\\_string\\_type](#) &\_\_s)
  - template<\_\_allocator\_like \_SAlloc>  
[basic\\_string](#)<\_CharT, \_Traits, \_SAlloc> [str](#) (const \_SAlloc &\_\_sa) const
  - template<\_\_allocator\_like \_SAlloc>  
requires (is\_same\_v<\_SAlloc, \_Alloc>)  
void [str](#) (const [basic\\_string](#)<\_CharT, \_Traits, \_SAlloc> &\_\_s)
  - void [swap](#) ([basic\\_ostringstream](#) &\_\_rhs)
  - pos\_type [tellp](#) ()
  - [basic\\_ostream](#)<\_CharT, \_Traits> \* [tie](#) () const
  - [basic\\_ostream](#)<\_CharT, \_Traits> \* [tie](#) ([basic\\_ostream](#)<\_CharT, \_Traits> \*\_\_tiestr)
  - void [unsetf](#) (fmtflags \_\_mask)
  - [basic\\_string\\_view](#)<char\_type, traits\_type> [view](#) () const noexcept
  - char\_type [widen](#) (char \_\_c) const
  - [streamsize](#) [width](#) () const
  - [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type` & `put` (`char_type __c`)
- `__ostream_type` & `write` (`const char_type *__s`, `streamsize __n`)
- `operator bool` () const
- `bool operator!` () const

### Static Public Member Functions

- static `bool sync_with_stdio` (`bool __sync=true`)
- static `int xalloc` () throw ()

### Public Attributes

- class `__attribute__((__abi_tag__("cxx11")))` failure typedef `_ios_Fmtflags` `fmtflags`

### Static Public Attributes

- static const `openmode` `__noreplace`
- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`

- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) & \_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- template<typename \_ValueT>  
[\\_\\_ostream\\_type](#) & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) & \_\_rhs) noexcept
- void [init](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- void [move](#) ([basic\\_ios](#) && \_\_rhs)
- void [move](#) ([basic\\_ios](#) & \_\_rhs)
- void [set\\_rdbuf](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- void [swap](#) ([basic\\_ios](#) & \_\_rhs) noexcept
- void [swap](#) ([basic\\_ostream](#) & \_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const [\\_\\_ctype\\_type](#) \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- [char\\_type](#) [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* [\\_M\\_num\\_get](#)
- const [\\_\\_num\\_put\\_type](#) \* [\\_M\\_num\\_put](#)
- [streamsize](#) [\\_M\\_precision](#)
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* [\\_M\\_streambuf](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [basic\\_ostream](#)<\_CharT, \_Traits> \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

### 5.229.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>  
class std::basic_ostringstream< _CharT, _Traits, _Alloc >
```

Controlling output for std::string.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

## 5.229.2 Member Typedef Documentation

`__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]
```

These are non-standard types.

`event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

`iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

`openmode`

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

### seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

## 5.229.3 Member Enumeration Documentation

### event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

## 5.229.4 Constructor & Destructor Documentation

### basic\_ostringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream () [inline]
```

Default constructor starts with an empty string buffer.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

### basic\_ostringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream (
    ios_base::openmode __mode) [inline], [explicit]
```

Starts with an empty string buffer.

#### Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

`ios_base::out` is automatically included in `mode`.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

**basic\_ostringstream()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream (
    const __string_type & __str,
    ios_base::openmode __mode = ios_base::out) [inline], [explicit]
```

Starts with an existing string buffer.

**Parameters**

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

`ios_base::out` is automatically included in *mode*.

Initializes *sb* using *str* and *mode*, and passes *&sb* to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

**~basic\_ostringstream()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::~~basic_ostringstream () [inline]
```

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

**5.229.5 Member Function Documentation****\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get_date()`, `std::time_get<_CharT, _Inlter>::do_get_monthname()`, `std::time_get<_CharT, _Inlter>::do_get_weekday()`, `std::time_get<_CharT, _Inlter>::do_get_year()`, `std::num_put<_CharT, _Outlter>::do_put()`, `std::time_put<_CharT, _Outlter>::do_put()`, `std::time_get<_CharT, _Inlter>::get()`, and `std::time_put<_CharT, _Outlter>::put()`.

**bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.



**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

## Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::exceptions\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::unget\(\)](#), and [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#).

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

## Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

## Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

**exceptions()** [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

## Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions()** [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Referenced by `std::basic_ios< char_type, traits_type >::operator bool()`, `std::basic_ios< char_type, traits_type >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellg()`, and `std::regex_traits< _Ch_type >::value()`.

**fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by `copyfmt()`, `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()()`, and `std::subtract_with_carry_engine`

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::fill (
    char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::subtract_with_carry_engine<uint_fast32_t>::do_put()`, `std::chrono::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::subtract_with_carry_engine<uint_fast32_t>::do_put()`.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

References [fmtflags](#).

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::pubsync\(\)](#).

## getloc()

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::money\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::do\\_get\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

## good()

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

## imbue()

```
template<typename _CharT, typename _Traits>
```

```
locale std::basic_ios<_CharT, _Traits>::imbue (
    const locale & __loc) [inherited]
```

Moves to a new locale.

### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::imbue\(\)](#), and [rdbuf\(\)](#).

Referenced by [std::chrono::operator<<\(\)](#).

## init()

```
template<typename _CharT, typename _Traits>
```

```
void std::basic_ios<_CharT, _Traits>::init (
    basic_streambuf<_CharT, _Traits> * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::basic\\_ios\(\)](#), and [std::basic\\_ostream<char\\_type, traits\\_type>::basic\\_ostream\(\)](#).

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [iword\(\)](#).

Referenced by [iword\(\)](#).

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __c,
    char __default) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator"!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(* __pf ) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(* __pf ) (__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

**operator<<() [4/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

## Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_↔</code> | A variable of pointer type. |
| <code>_p</code> |                             |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|                 |                                            |
|-----------------|--------------------------------------------|
| <code>↔</code>  | A variable of builtin floating point type. |
| <code>_↔</code> |                                            |
| <code>↔</code>  |                                            |
| <code>_↔</code> |                                            |
| <code>f</code>  |                                            |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    float __f) [inline], [inherited]
```

Floating point arithmetic inserters.



**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| $\leftrightarrow$    | A variable of builtin floating point type. |
| $\_ \leftrightarrow$ |                                            |
| $\leftrightarrow$    |                                            |
| $\_ \leftrightarrow$ |                                            |
| <i>f</i>             |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| $\_ \leftrightarrow$ | A variable of builtin integral type. |
| $\_ n$               |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [9/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(* __pf ) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `io manip` header.

**operator<<() [10/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| $\_ \leftrightarrow$ | A variable of builtin integral type. |
| $\_ n$               |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↔        | A variable of builtin floating point type. |
| ↔        |                                            |
| ↔        |                                            |
| ↔        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↔        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    short __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↔        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator<\\_CharT, \\_Traits>::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put<\\_CharT, \\_Outiter>::put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<()** [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| $\leftrightarrow$ | A variable of builtin integral type. |
| $n$               |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| $\leftrightarrow$ | A variable of builtin integral type. |
| $n$               |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| $\leftrightarrow$ | A variable of builtin integral type. |
| $n$               |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (
    char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits](#)

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc>::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

**rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic\_ios<\_CharT, \_Traits>::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::bad\(\)](#), [clear\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::eof\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::fail\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::good\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

**register\_callback()**

```
void std::ios\_base::register\_callback (
    event\_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

References [fmtflags](#).

**seekp()** [1/2]

```
template<typename _CharT, typename _Traits>
basic\_ostream<\_CharT, \_Traits> & std::basic\_ostream<\_CharT, \_Traits>::seekp (
    off\_type __off,
    ios\_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekp() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    pos_type __pos) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf() [1/2]**

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf() [2/2]**

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
    iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::getline()`, `std::getline()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, and `std::ws()`.

**str() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_ostringstream<_CharT, _Traits, _Alloc>::str () const & [inline],
[nodiscard]
```

Copying out the string buffer.

**Returns**

`rdbuf() -> str()`

**str() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_ostringstream<_CharT, _Traits, _Alloc>::str (
    const __string_type & __s) [inline]
```

Setting a new buffer.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Calls `rdbuf() -> str(s)`.



**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (  
    bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tell()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp () [inherited]
```

Getting the current write position.

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

References `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`

**tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `copyfmt()`.

**tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie (
    basic_ostream<_CharT, _Traits> * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

## Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References [fmtflags](#).

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::basic_ios< char_type, traits_type >::fill()`, `std::getline()`, `std::getline()`, `std::subtract_with_carry_engine< uint_fast32_t, std::tr2::operator>>()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::seed()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

## Returns

The previous value of width().

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n) [inherited]
```

Character string insertion.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

## Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

## Note

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static], [inherited]
```

Access to unique indices.

## Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.229.6 Member Data Documentation

#### adjustfield

const `fmtflags` `std::ios_base::adjustfield` [static], [inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### app

const `openmode` `std::ios_base::app` [static], [inherited]

Seek to end before each write.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

#### ate

const `openmode` `std::ios_base::ate` [static], [inherited]

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

#### badbit

const `iostate` `std::ios_base::badbit` [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char_type, traits_type >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_ostream< char_type, traits_type >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< char_type, traits_type >::ws()`, and `std::ws()`.

#### basefield

const `fmtflags` `std::ios_base::basefield` [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

#### beg

const `seekdir` `std::ios_base::beg` [static], [inherited]

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

#### binary

const `openmode` `std::ios_base::binary` [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`.

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

**eofbit**

```
const istate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<char_type, traits_type>::getline()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<char_type, traits_type>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<char_type, traits_type>::seekg()`, and `std::ws()`.

**failbit**

```
const istate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<char_type, traits_type>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, and `std::basic_ostream<_CharT, _Traits>::seekp()`.

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::readsome\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), and [std::ws\(\)](#).

**hex**

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::hex\(\)](#).

**in**

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Referenced by [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::showmanyc\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::underflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::xsgetn\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#).

**internal**

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

**left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.) Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, and `std::left()`.

### **oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by `std::oct()`.

### **out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

### **right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Referenced by `std::right()`.

### **scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by `std::hexfloat()`, and `std::scientific()`.

### **showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, `std::noshowbase()`, and `std::showbase()`.

### **showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

### **showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by `std::noshowpos()`, and `std::showpos()`.

### **skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by `std::noskipws()`, `std::__detail::operator>>()`, and `std::skipws()`.



**trunc**

const [openmode](#) std::ios\_base::trunc [static], [inherited]  
 Truncate an existing stream when opening. Default for ofstream.

**unitbuf**

const [fmtflags](#) std::ios\_base::unitbuf [static], [inherited]  
 Flushes output after each output operation.  
 Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

const [fmtflags](#) std::ios\_base::uppercase [static], [inherited]  
 Replaces certain lowercase letters with their uppercase equivalents in generated output.  
 Referenced by [std::num\\_put< \\_CharT, \\_Outiter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).  
 The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)

**5.230 std::basic\_regex< \_Ch\_type, \_Rx\_traits > Class Template Reference**

```
#include <regex>
```

**Public Types**

- typedef [regex\\_constants::syntax\\_option\\_type](#) **flag\_type**
- typedef traits\_type::locale\_type **locale\_type**
- typedef traits\_type::string\_type **string\_type**
- typedef \_Rx\_traits **traits\_type**
- typedef \_Ch\_type **value\_type**

**Public Member Functions**

- [basic\\_regex](#) () noexcept
- template<typename \_FwdIter>  
[basic\\_regex](#) (\_FwdIter \_\_first, \_FwdIter \_\_last, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) ([basic\\_regex](#) &&\_\_rhs) noexcept=default
- [basic\\_regex](#) (const \_Ch\_type \*\_\_p, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) (const \_Ch\_type \*\_\_p, std::size\_t \_\_len, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) (const [basic\\_regex](#) &\_\_rhs)=default
- template<typename \_Ch\_traits, typename \_Ch\_alloc>  
[basic\\_regex](#) (const [std::basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc > &\_\_s, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) ([initializer\\_list](#)< \_Ch\_type > \_\_l, [flag\\_type](#) \_\_f=ECMAScript)
- [~basic\\_regex](#) ()
- template<typename \_InputIterator>  
[basic\\_regex](#) & [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [flag\\_type](#) \_\_flags=ECMAScript)
- [basic\\_regex](#) & [assign](#) ([basic\\_regex](#) &&\_\_rhs) noexcept
- [basic\\_regex](#) & [assign](#) (const \_Ch\_type \*\_\_p, [flag\\_type](#) \_\_flags=ECMAScript)
- [basic\\_regex](#) & [assign](#) (const \_Ch\_type \*\_\_p, size\_t \_\_len, [flag\\_type](#) \_\_flags=ECMAScript)
- [basic\\_regex](#) & [assign](#) (const [basic\\_regex](#) &\_\_rhs) noexcept
- template<typename \_Ch\_traits, typename \_Alloc>  
[basic\\_regex](#) & [assign](#) (const [basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Alloc > &\_\_s, [flag\\_type](#) \_\_flags=ECMAScript)

- `basic_regex` & `assign` (`initializer_list`< `_Ch_type` > `__l`, `flag_type` `__flags`=ECMAScript)
- `flag_type` `flags` () const noexcept
- `locale_type` `getloc` () const noexcept
- `locale_type` `imbue` (`locale_type` `__loc`)
- unsigned int `mark_count` () const noexcept
- `basic_regex` & `operator=` (`basic_regex` &&)=default
- `basic_regex` & `operator=` (const `_Ch_type` \*`__p`)
- `basic_regex` & `operator=` (const `basic_regex` &)=default
- template<typename `_Ch_traits`, typename `_Alloc`>  
`basic_regex` & `operator=` (const `basic_string`< `_Ch_type`, `_Ch_traits`, `_Alloc` > &`__s`)
- `basic_regex` & `operator=` (`initializer_list`< `_Ch_type` > `__l`)
- void `swap` (`basic_regex` &`__rhs`) noexcept

## Static Public Attributes

### Constants

*std* [28.8.1](1)

- static constexpr `flag_type` `icase`
- static constexpr `flag_type` `nosubs`
- static constexpr `flag_type` `optimize`
- static constexpr `flag_type` `collate`
- static constexpr `flag_type` `ECMAScript`
- static constexpr `flag_type` `basic`
- static constexpr `flag_type` `extended`
- static constexpr `flag_type` `awk`
- static constexpr `flag_type` `grep`
- static constexpr `flag_type` `egrep`
- static constexpr `flag_type` `multiline`

## Friends

- template<typename `_Bp`, typename `_Ap`, typename `_Cp`, typename `_Rp`>  
bool `__detail::regex_algo_impl` (`_Bp`, `_Bp`, `match_results`< `_Bp`, `_Ap` > &, const `basic_regex`< `_Cp`, `_Rp` > &, `regex_constants::match_flag_type`, `__detail::_RegexExecutorPolicy`, bool)
- template<typename, typename, typename, bool>  
class `__detail::_Executor`

## Related Symbols

(Note that these are not member symbols.)

- template<typename `_Ch_type`, typename `_Rx_traits`>  
void `swap` (`basic_regex`< `_Ch_type`, `_Rx_traits` > &`__lhs`, `basic_regex`< `_Ch_type`, `_Rx_traits` > &`__rhs`) noexcept

### 5.230.1 Detailed Description

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::basic_regex<_Ch_type, _Rx_traits>
```

A regular expression.

Specializations of this class template represent regular expressions constructed from sequences of character type `__Ch_type`. Use the `std::regex` typedef for `std::basic_regex<char>`.

A character sequence passed to the constructor will be parsed according to the chosen grammar, and used to create a state machine representing the regular expression. The `regex` object can then be passed to algorithms such as `std::regex_match` to match sequences of characters.

The `syntax_option_type` flag passed to the constructor selects from one of the supported regular expression grammars. The default is `ECMAScript` and the others are `basic`, `extended`, `awk`, `grep`, and `egrep`, which are variations on POSIX regular expressions.

Since

C++11

## 5.230.2 Constructor & Destructor Documentation

### `basic_regex()` [1/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex () [inline], [noexcept]
```

Constructs a basic regular expression that does not match any character sequence.

### `basic_regex()` [2/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    const _Ch_type * __p,
    flag_type __f = ECMAScript) [inline], [explicit]
```

Constructs a basic regular expression from the sequence `[__p, __p + char_traits<_Ch_type>::length(__p))` interpreted according to the flags in `__f`.

#### Parameters

|                  |                                                                                             |
|------------------|---------------------------------------------------------------------------------------------|
| <code>__p</code> | A pointer to the start of a C-style null-terminated string containing a regular expression. |
| <code>__f</code> | Flags indicating the syntax rules and options.                                              |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__p</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

### `basic_regex()` [3/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    const _Ch_type * __p,
    std::size_t __len,
    flag_type __f = ECMAScript) [inline]
```

Constructs a basic regular expression from the sequence `[p, p + len)` interpreted according to the flags in `f`.

#### Parameters

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| <code>__p</code>   | A pointer to the start of a string containing a regular expression. |
| <code>__len</code> | The length of the string containing the regular expression.         |
| <code>__f</code>   | Flags indicating the syntax rules and options.                      |

## Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__p</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

**basic\_regex()** [4/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex<_Ch_type, _Rx_traits>::basic_regex (
    const basic_regex<_Ch_type, _Rx_traits> & __rhs) [default]
```

Copy-constructs a basic regular expression.

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__rhs</code> | A regex object. |
|--------------------|-----------------|

**basic\_regex()** [5/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex<_Ch_type, _Rx_traits>::basic_regex (
    basic_regex<_Ch_type, _Rx_traits> && __rhs) [default], [noexcept]
```

Move-constructs a basic regular expression.

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__rhs</code> | A regex object. |
|--------------------|-----------------|

**basic\_regex()** [6/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Ch_alloc>
std::basic_regex<_Ch_type, _Rx_traits>::basic_regex (
    const std::basic_string<_Ch_type, _Ch_traits, _Ch_alloc> & __s,
    flag_type __f = ECMAScript) [inline], [explicit]
```

Constructs a basic regular expression from the string `s` interpreted according to the flags in `f`.

## Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__s</code> | A string containing a regular expression.      |
| <code>__f</code> | Flags indicating the syntax rules and options. |

## Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__s</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

**basic\_regex()** [7/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _FwdIter>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    _FwdIter __first,
    _FwdIter __last,
    flag_type __f = ECMAScript) [inline]
```

Constructs a basic regular expression from the range [first, last) interpreted according to the flags in f.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | The start of a range containing a valid regular expression. |
| <code>__last</code>  | The end of a range containing a valid regular expression.   |
| <code>__f</code>     | The format flags of the regular expression.                 |

**Exceptions**

|                    |                                                                                      |
|--------------------|--------------------------------------------------------------------------------------|
| <i>regex_error</i> | if [ <code>__first</code> , <code>__last</code> ) is not a valid regular expression. |
|--------------------|--------------------------------------------------------------------------------------|

**basic\_regex()** [8/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    initializer_list< _Ch_type > __l,
    flag_type __f = ECMAScript) [inline]
```

Constructs a basic regular expression from an initializer list.

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__l</code> | The initializer list.                       |
| <code>__f</code> | The format flags of the regular expression. |

**Exceptions**

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <i>regex_error</i> | if <code>__l</code> is not a valid regular expression. |
|--------------------|--------------------------------------------------------|

**~basic\_regex()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::~~basic_regex () [inline]
```

Destroys a basic regular expression.

### 5.230.3 Member Function Documentation

#### assign() [1/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _InputIterator>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::assign (
    _InputIterator __first,
    _InputIterator __last,
    flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object.

##### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | The start of a range containing a valid regular expression. |
| <code>__last</code>  | The end of a range containing a valid regular expression.   |
| <code>__flags</code> | Syntax option flags.                                        |

##### Exceptions

|                          |                                                                                                                                                                               |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if p does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged. |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### assign() [2/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::assign (
    basic_regex<_Ch_type, _Rx_traits > && __rhs) [inline], [noexcept]
```

Move-assigns one regular expression to another.

##### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

#### assign() [3/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::assign (
    const _Ch_type * __p,
    flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

##### Parameters

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| <code>__p</code>     | A pointer to a C-style null-terminated string containing a regular expression pattern. |
| <code>__flags</code> | Syntax option flags.                                                                   |

## Exceptions

|                    |                                                                                                                                                                                                      |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>regex_error</i> | if <code>__p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**assign()** [4/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
    const _Ch_type * __p,
    size_t __len,
    flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

## Parameters

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| <code>__p</code>     | A pointer to a C-style string containing a regular expression pattern. |
| <code>__len</code>   | The length of the regular expression pattern string.                   |
| <code>__flags</code> | Syntax option flags.                                                   |

## Exceptions

|                    |                                                                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>regex_error</i> | if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**assign()** [5/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
    const basic_regex< _Ch_type, _Rx_traits > & __rhs) [inline], [noexcept]
```

Assigns one regular expression to another.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

Referenced by `std::basic_regex< char >::basic_regex()`, `std::basic_regex< char >::assign()`, `std::basic_regex< char >::operator=()`, `std::basic_regex< char >::operator=()`, and `std::basic_regex< char >::operator=()`.

**assign()** [6/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
    const basic_string< _Ch_type, _Ch_traits, _Alloc > & __s,
    flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__s</code>     | A string containing a regular expression pattern. |
| <code>__flags</code> | Syntax option flags.                              |

## Exceptions

|                          |                                                                                                                                                                                                      |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>__s</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**assign()** [7/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits>::assign (
    initializer_list<_Ch_type> __l,
    flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object.

## Parameters

|                      |                                                        |
|----------------------|--------------------------------------------------------|
| <code>__l</code>     | An initializer list representing a regular expression. |
| <code>__flags</code> | Syntax option flags.                                   |

## Exceptions

|                          |                                                                                                                                                                                              |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>__l</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged. |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**flags()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
flag_type std::basic_regex<_Ch_type, _Rx_traits>::flags () const [inline], [noexcept]
```

Gets the flags used to construct the regular expression or in the last call to `assign()`.

**getloc()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex<_Ch_type, _Rx_traits>::getloc () const [inline], [noexcept]
```

Gets the locale currently imbued in the regular expression object.

**imbue()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex<_Ch_type, _Rx_traits>::imbue (
    locale_type __loc) [inline]
```

Imbues the regular expression object with the given locale.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__loc</code> | A locale. |
|--------------------|-----------|



**mark\_count()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
unsigned int std::basic_regex<_Ch_type, _Rx_traits >::mark_count () const [inline], [noexcept]
```

Gets the number of marked subexpressions within the regular expression.

**operator=()** [1/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    basic_regex<_Ch_type, _Rx_traits > && ) [default]
```

Move-assigns one regular expression to another.

**operator=()** [2/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    const _Ch_type * __p) [inline]
```

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

**Parameters**

|                  |                                                                                             |
|------------------|---------------------------------------------------------------------------------------------|
| <code>__p</code> | A pointer to the start of a null-terminated C-style string containing a regular expression. |
|------------------|---------------------------------------------------------------------------------------------|

**operator=()** [3/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    const basic_regex<_Ch_type, _Rx_traits > & ) [default]
```

Assigns one regular expression to another.

**operator=()** [4/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    const basic_string<_Ch_type, _Ch_traits, _Alloc > & __s) [inline]
```

Replaces a regular expression with a new one constructed from a string.

**Parameters**

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__s</code> | A pointer to a string containing a regular expression. |
|------------------|--------------------------------------------------------|

**operator=()** [5/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    initializer_list<_Ch_type > __l) [inline]
```

Replaces a regular expression with a new one constructed from an initializer list.

## Parameters

|                 |                       |
|-----------------|-----------------------|
| <code>↩</code>  | The initializer list. |
| <code>_↩</code> |                       |
| <code>↩</code>  |                       |
| <code>_↩</code> |                       |
| <code>/</code>  |                       |

## Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__l</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

**swap()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
void std::basic_regex<_Ch_type, _Rx_traits>::swap (
    basic_regex<_Ch_type, _Rx_traits> & __rhs) [inline], [noexcept]
```

Swaps the contents of two regular expression objects.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

The documentation for this class was generated from the following file:

- [regex.h](#)

**5.231 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference**

```
#include <streambuf>
```

Inheritance diagram for `std::basic_streambuf< _CharT, _Traits >`:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_Traits` [traits\\_type](#)
- typedef `traits_type::int_type` [int\\_type](#)
- typedef `traits_type::pos_type` [pos\\_type](#)
- typedef `traits_type::off_type` [off\\_type](#)
- typedef `basic_streambuf< char_type, traits_type >` [\\_\\_streambuf\\_type](#)

## Public Member Functions

- virtual `~basic_streambuf()`
- `locale getloc()` const
- `streamsize in_avail()`
- `locale pubimbue(const locale &__loc)`
- `int_type sbumpc()`
- `int_type sgetc()`
- `streamsize sgetn(char_type *__s, streamsize __n)`
- `int_type snextc()`
- `int_type sputbackc(char_type __c)`
- `int_type sputc(char_type __c)`
- `streamsize sputn(const char_type *__s, streamsize __n)`
- `int_type sungetc()`

- [basic\\_streambuf \\* pubsetbuf](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [pos\\_type pubseekoff](#) ([off\\_type](#) \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [pos\\_type pubseekpos](#) ([pos\\_type](#) \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [int pubsync](#) ()

### Protected Member Functions

- [basic\\_streambuf](#) ()
  - **basic\_streambuf** (const [basic\\_streambuf](#) &)
  - void [\\_\\_safe\\_gbump](#) ([streamsize](#) \_\_n)
  - void [\\_\\_safe\\_pbump](#) ([streamsize](#) \_\_n)
  - void [gbump](#) (int \_\_n)
  - virtual void [imbue](#) (const [locale](#) & \_\_loc)
  - [basic\\_streambuf](#) & **operator=** (const [basic\\_streambuf](#) &)
  - virtual [int\\_type](#) [overflow](#) ([int\\_type](#) \_\_c=[traits\\_type::eof](#)())
  - virtual [int\\_type](#) [pbackfail](#) ([int\\_type](#) \_\_c=[traits\\_type::eof](#)())
  - void [pbump](#) (int \_\_n)
  - virtual [pos\\_type](#) [seekoff](#) ([off\\_type](#), [ios\\_base::seekdir](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
  - virtual [pos\\_type](#) [seekpos](#) ([pos\\_type](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
  - virtual [basic\\_streambuf](#)< [char\\_type](#), [\\_Traits](#) > \* [setbuf](#) ([char\\_type](#) \*, [streamsize](#))
  - void [setg](#) ([char\\_type](#) \* \_\_gbeg, [char\\_type](#) \* \_\_gnext, [char\\_type](#) \* \_\_gend)
  - void [setp](#) ([char\\_type](#) \* \_\_pbeg, [char\\_type](#) \* \_\_pend)
  - virtual [streamsize](#) [showmanyc](#) ()
  - void **swap** ([basic\\_streambuf](#) & \_\_sb)
  - virtual [int](#) [sync](#) ()
  - virtual [int\\_type](#) [uflow](#) ()
  - virtual [int\\_type](#) [underflow](#) ()
  - virtual [streamsize](#) [xsgetn](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
  - virtual [streamsize](#) [xspn](#) (const [char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- 
- [char\\_type](#) \* [eback](#) () const
  - [char\\_type](#) \* [gptr](#) () const
  - [char\\_type](#) \* [egptr](#) () const
- 
- [char\\_type](#) \* [pbase](#) () const
  - [char\\_type](#) \* [pptr](#) () const
  - [char\\_type](#) \* [epptr](#) () const

### Protected Attributes

- [locale](#) [\\_M\\_buf\\_locale](#)
- [char\\_type](#) \* [\\_M\\_in\\_beg](#)
- [char\\_type](#) \* [\\_M\\_in\\_cur](#)
- [char\\_type](#) \* [\\_M\\_in\\_end](#)
- [char\\_type](#) \* [\\_M\\_out\\_beg](#)
- [char\\_type](#) \* [\\_M\\_out\\_cur](#)
- [char\\_type](#) \* [\\_M\\_out\\_end](#)

## Friends

- `template<bool _IsMove, typename _CharT2>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)`
- `streamsize __copy_streambufs_eof(basic_streambuf * __sbin, basic_streambuf * __sbout, bool & __ineof)`
- `void __istream_extract(istream &, char *, streamsize)`
- `template<typename _CharT2, typename _Distance>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type advance(istreambuf_iterator< _CharT2 > &, _Distance)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`
- `class basic_ostream< char_type, traits_type >`
- `template<typename _CharT2>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type find`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)`
- `template<typename _CharT2, typename _Traits2, typename _Alloc>`  
`basic_istream< _CharT2, _Traits2 > & getline(basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2,`  
`_Traits2, _Alloc > &, _CharT2)`
- `class istreambuf_iterator< char_type, traits_type >`
- `template<typename _CharT2, typename _Traits2, typename _Alloc>`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, basic_string<`  
`_CharT2, _Traits2, _Alloc > &)`
- `class ostreambuf_iterator< char_type, traits_type >`

### 5.231.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_streambuf< _CharT, _Traits >
```

The actual work of input and output (interface).

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:

- The controlled input sequence can be not readable.
- The controlled output sequence can be not writable.
- The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
- The controlled sequences can support operations *directly* to or from associated sequences.
- The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.

2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
  - the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
  - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
  - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
  - If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
  - If *xnext* is not a null pointer and *xnext* < *xend* for an output sequence, then a *write position* is available. In this case, *\*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
  - If *xnext* is not a null pointer and *xbeg* < *xnext* for an input sequence, then a *putback position* is available. In this case, *xnext*[-1] shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
  - If *xnext* is not a null pointer and *xnext* < *xend* for an input sequence, then a *read position* is available. In this case, *\*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

### 5.231.2 Member Typedef Documentation

#### `__streambuf_type`

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf<_CharT, _Traits>::__↵
streambuf_type
```

This is a non-standard type.

#### `char_type`

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_streambuf<_CharT, _Traits>::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

#### `int_type`

```
template<typename _CharT, typename _Traits>
typedef traits_type::int_type std::basic_streambuf<_CharT, _Traits>::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

#### `off_type`

```
template<typename _CharT, typename _Traits>
typedef traits_type::off_type std::basic_streambuf<_CharT, _Traits>::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**pos\_type**

```
template<typename _CharT, typename _Traits>
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**traits\_type**

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**5.231.3 Constructor & Destructor Documentation****~basic\_streambuf()**

```
template<typename _CharT, typename _Traits>
virtual std::basic_streambuf< _CharT, _Traits >::~~basic_streambuf () [inline], [virtual]
```

Destructor deallocates no buffer space.

**basic\_streambuf()**

```
template<typename _CharT, typename _Traits>
std::basic_streambuf< _CharT, _Traits >::basic_streambuf () [inline], [protected]
```

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the `basic_streambuf` class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

**5.231.4 Member Function Documentation****eback()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**egptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::xsgetn()`, and `xsgetn()`.

**epptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::xsputn()`, and `xsputn()`.

**gbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
    int __n) [inline], [protected]
```

Moving the read position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Referenced by `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline]
```

Locale access.



## Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

## gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

## imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
    const locale & __loc) [inline], [protected], [virtual]
```

Changes translations.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, and `std::basic_filebuf< wchar_t >`.

## in\_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline]
```

Looking ahead into the stream.

## Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**overflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::overflow (
    int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, traits\\_type>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<wchar\\_t>](#), [std::basic\\_filebuf<wchar\\_t>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#), [std::basic\\_stringbuf<char>](#), [std::basic\\_stringbuf<char>](#), [std::basic\\_stringbuf<wchar\\_t>](#), [std::basic\\_stringbuf<wchar\\_t>](#), and [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>](#).

Referenced by [xsputn\(\)](#).

**pbackfail()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
    int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, traits\\_type>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<wchar\\_t>](#), [std::basic\\_filebuf<wchar\\_t>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#), [std::basic\\_stringbuf<char>](#), [std::basic\\_stringbuf<char>](#), [std::basic\\_stringbuf<wchar\\_t>](#), and [std::basic\\_stringbuf<wchar\\_t>](#).

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::sync\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#).

**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
    int __n) [inline], [protected]
```

Moving the write position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#).

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::sync\(\)](#), and [xsputn\(\)](#).

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc) [inline]
```

Entry point for imbue().

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

## Returns

The previous locale.

Calls the derived `imbue(__loc)`.

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf<_CharT, _Traits>::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekoff` function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf<_CharT, _Traits>::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

**pubsetbuf()**

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf<_CharT, _Traits>::pubsetbuf (
    char_type * __s,
    streamsize __n) [inline]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

**pubsync()**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline]
```

Calls virtual sync function.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#).

**sbumpc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::getline\(\)](#).

**seekoff()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), [std::basic\\_filebuf< wchar\\_t >](#), [std::basic\\_filebuf< wchar\\_t >](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_stringbuf< char >](#), [std::basic\\_stringbuf< char >](#), [std::basic\\_stringbuf< wchar\\_t >](#), and [std::basic\\_stringbuf< wchar\\_t >](#).

**seekpos()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), [std::basic\\_filebuf< wchar\\_t >](#), [std::basic\\_filebuf< wchar\\_t >](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_stringbuf< char >](#), [std::basic\\_stringbuf< char >](#), [std::basic\\_stringbuf< wchar\\_t >](#), and [std::basic\\_stringbuf< wchar\\_t >](#).

**setbuf()**

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
    char_type * ,
    streamsizet ) [inline], [protected], [virtual]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

**Note**

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_stringbuf<char>`, and `std::basic_stringbuf<char>`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend) [inline], [protected]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
    char_type * __pbeg,
    char_type * __pend) [inline], [protected]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by `std::wbuffer_convert<_Codecv, _Elem, _Tr>::wbuffer_convert()`.

**sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc () [inline]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::ignore()`.

**sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
    char_type * __s,
    streamsize __n) [inline]
```

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

**showmanyc()**

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf<_CharT, _Traits>::showmanyc () [inline], [protected], [virtual]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1*

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return eof() but that they will return immediately.*

The standard adds that *the morphemes of showmanyc are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_filebuf< wchar_t >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_stringbuf< char >`, `std::basic_stringbuf< char >`, `std::basic_stringbuf< wchar_t >`, and `std::basic_stringbuf< wchar_t >`.

**snextc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline]
Getting the next character.
```

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< char_type, traits_type >::ignore()`, `std::basic_istream< char_type, traits_type >::operator>>()`, and `std::basic_istream< char_type, traits_type >::putback()`.

**sputbackc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
    char_type __c) [inline]
Pushing characters back into the input stream.
```

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
    char_type __c) [inline]
Entry point for all single-character output functions.
```



**Parameters**

|                          |                        |
|--------------------------|------------------------|
| $\leftrightarrow$<br>__c | A character to output. |
|--------------------------|------------------------|

**Returns**

\_\_c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores \_\_c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↔ __c)`.

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
    const char_type * __s,
    streamsize __n) [inline]
```

Entry point for all single-character output functions.

**Parameters**

|                          |                     |
|--------------------------|---------------------|
| $\leftrightarrow$<br>__s | A buffer read area. |
| $\leftrightarrow$<br>__n | A count.            |

One of two public output functions.

Returns `xsgputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc () [inline]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream<char_type, traits_type>::sentry::sentry()`.

**sync()**

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf<_CharT, _Traits>::sync () [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_filebuf< wchar_t >`, and `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`.

**uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual]
Fetches more data from the controlled sequence.
```

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Referenced by `xsgetn()`.

**underflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::underflow () [inline], [protected],
[virtual]
Fetches more data from the controlled sequence.
```

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_filebuf< wchar_t >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_stringbuf< char >`, `std::basic_stringbuf< char >`, `std::basic_stringbuf< wchar_t >`, `std::basic_stringbuf< wchar_t >`, and `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`.

**xsgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
    char_type * __s,
    streamsize __n) [protected], [virtual]
Multiple character extraction.
```

## Parameters

|                                  |                                         |
|----------------------------------|-----------------------------------------|
| <a href="#"><code>__s</code></a> | A buffer area.                          |
| <a href="#"><code>__n</code></a> | Maximum number of characters to assign. |

## Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), and [std::basic\\_filebuf<wchar\\_t>](#).  
References [egptr\(\)](#), [gptr\(\)](#), [std::min\(\)](#), and [uflow\(\)](#).

**xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xsputn (
    const char_type * __s,
    streamsize __n)    [protected], [virtual]
```

Multiple character insertion.

## Parameters

|                                  |                                        |
|----------------------------------|----------------------------------------|
| <a href="#"><code>__s</code></a> | A buffer area.                         |
| <a href="#"><code>__n</code></a> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), and [std::basic\\_filebuf<wchar\\_t>](#).  
References [epptr\(\)](#), [std::min\(\)](#), [overflow\(\)](#), and [pptr\(\)](#).

**5.231.5 Member Data Documentation****`_M_buf_locale`**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale    [protected]
```

Current locale setting.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::basic\\_filebuf\(\)](#).

**`_M_in_beg`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected]
```

Start of get area.

**`_M_in_cur`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected]
```

Current read area.

**`_M_in_end`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected]
```

End of get area.

**`_M_out_beg`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected]
```

Start of put area.

**`_M_out_cur`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected]
```

Current put area.

**`_M_out_end`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected]
```

End of put area.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [streambuf](#)
- [streambuf.tcc](#)

**5.232 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference**

```
#include <string>
```

Inheritance diagram for `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, basic_string >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::iterator, basic_string >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- **basic\_string** (`_Base` && `__base`) noexcept
- template<typename `_InputIterator`>  
  **basic\_string** (`_InputIterator` `__begin`, `_InputIterator` `__end`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (`basic_string` &&)=default
- **basic\_string** (`basic_string` && `__s`, const `_Allocator` & `__a`) noexcept(`std::is_nothrow_constructible< _Base, _Base, const _Allocator & >::value`)
- **basic\_string** (const `_Allocator` & `__a`) noexcept
- **basic\_string** (const `_Base` & `__base`)
- **basic\_string** (const `_CharT` \* `__s`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (const `_CharT` \* `__s`, `size_type` `__n`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (const `basic_string` &)=default
- **basic\_string** (const `basic_string` & `__s`, const `_Allocator` & `__a`)
- **basic\_string** (const `basic_string` & `__str`, `size_type` `__pos`, `size_type` `__n`=`_Base::npos`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (`size_type` `__n`, `_CharT` `__c`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (`std::initializer_list< _CharT >` `__l`, const `_Allocator` & `__a`=`_Allocator`())

- `template<typename _Operation>`  
`constexpr void __resize_and_overwrite (size_type __n, _Operation __op)`
- `template<typename _Operation>`  
`void __resize_and_overwrite (size_type __n, _Operation __op)`
- `const _Base & _M_base () const noexcept`
- `_Base & _M_base () noexcept`
- `template<typename _InputIterator>`  
`basic_string<_CharT, _Traits, _Alloc> & _M_replace_dispatch (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2, __false_type)`
- `template<typename _Iterator>`  
`_CharT * _S_construct (_Iterator __beg, _Iterator __end, const _Alloc & __a, forward_iterator_tag)`
- `template<typename _InputIterator>`  
`basic_string & append (_InputIterator __first, _InputIterator __last)`
- `basic_string & append (const _CharT * __s)`
- `basic_string & append (const _CharT * __s, size_type __n)`
- `basic_string & append (const basic_string & __str)`
- `basic_string & append (const basic_string & __str, size_type __pos, size_type __n)`
- `basic_string & append (size_type __n, _CharT __c)`
- `constexpr basic_string & append (const _CharT * __s, size_type __n)`
- `constexpr basic_string & append (const basic_string & __str)`
- `constexpr basic_string & append (const basic_string & __str, size_type __pos, size_type __n=npos)`
- `constexpr basic_string & append (initializer_list<_CharT> __l)`
- `constexpr basic_string & append (size_type __n, _CharT __c)`
- `template<typename _InputIterator>`  
`basic_string & assign (_InputIterator __first, _InputIterator __last)`
- `basic_string & assign (basic_string && __x) noexcept(noexcept(std::declval<_Base &>().assign(std::move(__x))))`
- `basic_string & assign (const _CharT * __s)`
- `basic_string & assign (const _CharT * __s, size_type __n)`
- `basic_string & assign (const basic_string & __str, size_type __pos, size_type __n)`
- `basic_string & assign (const basic_string & __x)`
- `basic_string & assign (size_type __n, _CharT __c)`
- `basic_string & assign (std::initializer_list<_CharT> __l)`
- `constexpr basic_string & assign (basic_string && __str) noexcept(_Alloc_traits::_S_nothrow_move())`
- `constexpr basic_string & assign (const _CharT * __s, size_type __n)`
- `constexpr basic_string & assign (const basic_string & __str)`
- `constexpr basic_string & assign (const basic_string & __str, size_type __pos, size_type __n=npos)`
- `constexpr basic_string & assign (size_type __n, _CharT __c)`
- `constexpr reference at (size_type __n)`
- `reference at (size_type __n)`
- `constexpr const_reference at (size_type __n) const`
- `const_reference at (size_type __n) const`
- `constexpr reference at (size_type __n)`
- `constexpr const_reference at (size_type __n) const`
- `constexpr const_reference back () const noexcept`
- `constexpr reference back () noexcept`
- `iterator begin ()`
- `const_iterator begin () const noexcept`
- `const _CharT * c_str () const noexcept`
- `constexpr size_type capacity () const noexcept`
- `const_iterator cbegin () const noexcept`

- `const_iterator cend ()` const noexcept
- void `clear ()`
- constexpr int `compare` (const `_CharT * __s`) const noexcept
- constexpr int `compare` (const `basic_string & __str`) const
- constexpr int `compare` (size\_type \_\_pos, size\_type \_\_n, const `basic_string & __str`) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n, const `basic_string & __str`) const
- constexpr int `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT * __s`) const
- constexpr int `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT * __s`, size\_type \_\_n2) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT * __s`, size\_type \_\_n2) const
- constexpr int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `_CharT * __s`) const
- constexpr int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `_CharT * __s`, size\_type \_\_n2) const
- constexpr int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string & __str`, size\_type \_\_pos2, size\_type \_\_n2=`npos`) const
- int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string & __str`, size\_type \_\_pos2, size\_type \_\_n2=`npos`) const
- constexpr int `compare` (size\_type \_\_pos, size\_type \_\_n, const `basic_string & __str`) const
- constexpr int `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT * __s`) const
- constexpr int `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT * __s`, size\_type \_\_n2) const
- constexpr int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string & __str`, size\_type \_\_pos2, size\_type \_\_n2=`npos`) const
- size\_type `copy` (`_CharT * __s`, size\_type \_\_n, size\_type \_\_pos=0) const
- constexpr size\_type `copy` (`_CharT * __s`, size\_type \_\_n, size\_type \_\_pos=0) const
- `const_reverse_iterator crbegin ()` const noexcept
- `const_reverse_iterator crend ()` const noexcept
- const `_CharT * data ()` const noexcept
- constexpr `_CharT * data ()` noexcept
- constexpr bool `empty ()` const noexcept
- `iterator end ()`
- `const_iterator end ()` const noexcept
- constexpr bool `ends_with` (`_CharT __x`) const noexcept
- bool `ends_with` (`_CharT __x`) const noexcept
- constexpr bool `ends_with` (`basic_string_view< _CharT, _Traits > __x`) const noexcept
- bool `ends_with` (`basic_string_view< _CharT, _Traits > __x`) const noexcept
- constexpr bool `ends_with` (const `_CharT * __x`) const noexcept
- bool `ends_with` (const `_CharT * __x`) const noexcept
- `iterator erase` (`__const_iterator __first`, `__const_iterator __last`)
- `iterator erase` (`__const_iterator __position`)
- `basic_string & erase` (size\_type \_\_pos=0, size\_type \_\_n=`Base::npos`)
- constexpr `iterator erase` (`__const_iterator __first`, `__const_iterator __last`)
- constexpr `iterator erase` (`__const_iterator __position`)
- `iterator erase` (`iterator __first`, `iterator __last`)
- `iterator erase` (`iterator __position`)
- constexpr `basic_string & erase` (size\_type \_\_pos=0, size\_type \_\_n=`npos`)
- constexpr size\_type `find` (`_CharT __c`, size\_type \_\_pos=0) const noexcept
- size\_type `find` (`_CharT __c`, size\_type \_\_pos=0) const noexcept
- constexpr size\_type `find` (const `_CharT * __s`, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find` (const `_CharT * __s`, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type `find` (const `_CharT * __s`, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type `find` (const `_CharT * __s`, size\_type \_\_pos=0) const noexcept
- constexpr size\_type `find` (const `_CharT * __s`, size\_type \_\_pos=0) const noexcept
- constexpr size\_type `find` (const `basic_string & __str`, size\_type \_\_pos=0) const noexcept

- [illegible]



- constexpr size\_type [find\\_last\\_of](#) (const [basic\\_string](#) & \_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [find\\_last\\_of](#) (const [basic\\_string](#) & \_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- constexpr size\_type [find\\_last\\_of](#) ( \_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- constexpr size\_type [find\\_last\\_of](#) (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type [find\\_last\\_of](#) (const \_CharT \* \_\_s, size\_type \_\_pos=[npos](#)) const noexcept
- constexpr size\_type [find\\_last\\_of](#) (const [basic\\_string](#) & \_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- constexpr const\_reference [front](#) () const noexcept
- constexpr reference [front](#) () noexcept
- constexpr allocator\_type [get\\_allocator](#) () const noexcept
- [iterator insert](#) ( [\\_\\_const\\_iterator](#) \_\_p, \_CharT \_\_c)
- template<typename \_InputIterator>  
[iterator insert](#) ( [\\_\\_const\\_iterator](#) \_\_p, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator insert](#) (const\_iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- [iterator insert](#) (const\_iterator \_\_p, std::initializer\_list< \_CharT > \_\_l)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s, size\_type \_\_n)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) & \_\_str, size\_type \_\_pos2, size\_type \_\_n)
- constexpr [iterator insert](#) ( [\\_\\_const\\_iterator](#) \_\_p, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::::RequireInputIter< \_InputIterator >>  
constexpr [iterator insert](#) (const\_iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- constexpr [iterator insert](#) (const\_iterator \_\_p, initializer\_list< \_CharT > \_\_l)
- constexpr [iterator insert](#) (const\_iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- [iterator insert](#) (iterator \_\_p, \_CharT \_\_c)
- template<class \_InputIterator>  
void [insert](#) (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void [insert](#) (iterator \_\_p, initializer\_list< \_CharT > \_\_l)
- void [insert](#) (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s, size\_type \_\_n)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) & \_\_str)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) & \_\_str, size\_type \_\_pos2, size\_type \_\_n ←  
n=[npos](#))
- constexpr size\_type [length](#) () const noexcept
- constexpr size\_type [max\\_size](#) () const noexcept
- [basic\\_string](#) & [operator+=](#) ( \_CharT \_\_c)
- [basic\\_string](#) & [operator+=](#) (const \_CharT \* \_\_s)
- [basic\\_string](#) & [operator+=](#) (const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [operator+=](#) (std::initializer\_list< \_CharT > \_\_l)
- constexpr [basic\\_string](#) & [operator+=](#) (const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [operator=](#) ( \_CharT \_\_c)
- [basic\\_string](#) & [operator=](#) ([basic\\_string](#) &&)=default
- [basic\\_string](#) & [operator=](#) (const \_CharT \* \_\_s)
- [basic\\_string](#) & [operator=](#) (const [basic\\_string](#) &)=default
- [basic\\_string](#) & [operator=](#) (std::initializer\_list< \_CharT > \_\_l)
- reference [operator\[\]](#) (size\_type \_\_pos)
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const noexcept
- constexpr reference [operator\[\]](#) (size\_type \_\_pos)
- constexpr const\_reference [operator\[\]](#) (size\_type \_\_pos) const noexcept

- `void pop_back ()`
- `void push_back (_CharT __c)`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin ()` `const noexcept`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend ()` `const noexcept`
- `template<typename _InputIterator>`
  - `basic_string & replace (__const_iterator __i1, __const_iterator __i2, _InputIterator __j1, _InputIterator __j2)`
- `basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s)`
- `basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s, size_type __n)`
- `basic_string & replace (__const_iterator __i1, __const_iterator __i2, const basic_string & __str)`
- `basic_string & replace (__const_iterator __i1, __const_iterator __i2, size_type __n, _CharT __c)`
- `basic_string & replace (__const_iterator __i1, __const_iterator __i2, std::initializer_list<_CharT> __l)`
- `basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s)`
- `basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)`
- `basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str)`
- `basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str, size_type __pos2, size_type __n2)`
- `constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, _CharT *__k1, _CharT *__k2)`
- `constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__k1, const _CharT *__k2)`
- `constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s)`
- `constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s, size_type __n)`
- `constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const basic_string & __str)`
- `constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const_iterator __k1, const_iterator __k2)`
- `constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, iterator __k1, iterator __k2)`
- `constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, size_type __n, _CharT __c)`
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`
  - `constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)`
- `constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, initializer_list<_CharT> __l)`
- `basic_string & replace (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)`
- `template<class _InputIterator>`
  - `basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)`
- `basic_string & replace (iterator __i1, iterator __i2, const basic_string & __str)`
- `basic_string & replace (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)`
- `basic_string & replace (iterator __i1, iterator __i2, initializer_list<_CharT> __l)`
- `basic_string & replace (iterator __i1, iterator __i2, iterator __k1, iterator __k2)`
- `basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)`
- `constexpr basic_string & replace (size_type __pos, size_type __n, const basic_string & __str)`
- `constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s)`
- `constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)`
- `constexpr basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `constexpr basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str, size_type __pos2, size_type __n2=npos)`
- `constexpr void reserve ()`
- `constexpr void reserve (size_type __res_arg)`

- void **reserve** (size\_type \_\_res\_arg)
- constexpr void **reserve** (size\_type \_\_res\_arg)
- void **resize** (size\_type \_\_n)
- void **resize** (size\_type \_\_n, \_CharT \_\_c)
- constexpr void **resize** (size\_type \_\_n)
- constexpr void **resize** (size\_type \_\_n, \_CharT \_\_c)
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=**npos**) const noexcept
- size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=**npos**) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **rfind** (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type **rfind** (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_s, size\_type \_\_pos=**\_Base::npos**) const
- constexpr size\_type **rfind** (const \_CharT \*\_\_s, size\_type \_\_pos=**npos**) const
- constexpr size\_type **rfind** (const **basic\_string** &\_\_str, size\_type \_\_pos=**npos**) const noexcept
- size\_type **rfind** (const **basic\_string** &\_\_str, size\_type \_\_pos=**npos**) const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=**npos**) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_s, size\_type \_\_pos=**npos**) const
- constexpr size\_type **rfind** (const **basic\_string** &\_\_str, size\_type \_\_pos=**npos**) const noexcept
- void **shrink\_to\_fit** () noexcept
- constexpr size\_type **size** () const noexcept
- constexpr bool **starts\_with** (\_CharT \_\_x) const noexcept
- bool **starts\_with** (\_CharT \_\_x) const noexcept
- constexpr bool **starts\_with** (basic\_string\_view< \_CharT, \_Traits > \_\_x) const noexcept
- bool **starts\_with** (basic\_string\_view< \_CharT, \_Traits > \_\_x) const noexcept
- constexpr bool **starts\_with** (const \_CharT \*\_\_x) const noexcept
- bool **starts\_with** (const \_CharT \*\_\_x) const noexcept
- **basic\_string** **substr** (size\_type \_\_pos=0, size\_type \_\_n=**\_Base::npos**) const
- constexpr **basic\_string** **substr** (size\_type \_\_pos=0, size\_type \_\_n=**npos**) const
- void **swap** (**basic\_string** &\_\_x) noexcept(/\*conditional \*/)
- constexpr void **swap** (**basic\_string** &\_\_s) noexcept

### Static Public Attributes

- static const size\_type **npos**

### Protected Member Functions

- constexpr void **\_M\_swap** (const **\_Safe\_container** &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class **::gnu\_debug::Safe\_iterator**

### 5.232.1 Detailed Description

template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>

class **::gnu\_debug::basic\_string**< \_CharT, \_Traits, \_Allocator >

Class std::basic\_string with safety/checking/debug instrumentation.

### 5.232.2 Member Function Documentation

#### `__resize_and_overwrite()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string<_CharT, _Traits, _Alloc>::__resize_and_overwrite (
    size_type __n,
    _Operation __op) [constexpr], [inherited]
```

Non-standard version of `resize_and_overwrite` for C++11 and above.

#### `__resize_and_overwrite()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string<_CharT, _Traits, _Alloc>::__resize_and_overwrite (
    size_type __n,
    _Operation __op) [inherited]
```

Non-standard version of `resize_and_overwrite` for C++11 and above.

#### `append()` [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::append (
    const _CharT * __s,
    size_type __n) [inline], [constexpr], [inherited]
```

Append a C substring.

##### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__s</code> | The C string to append.             |
| <code>__n</code> | The number of characters to append. |

##### Returns

Reference to this string.

#### `append()` [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::append (
    const basic_string<_CharT, _Traits, _Alloc> & __str) [inline], [constexpr], [inherited]
```

Append a string to this string.

##### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Referenced by `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::append()`, `std::__cxx11::collate<_CharT>::do_transform()`, `std::basic_string<_CharT>::operator+=()`, `std::basic_string<_CharT>::operator+=()`, `std::basic_string<_CharT>::operator+=()`, `std::operator>>()`, and `resize()`.

**append()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::append (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos,
    size_type __n = npos) [inline], [constexpr], [inherited]
```

Append a substring.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__str</code> | The string to append.                                       |
| <code>__pos</code> | Index of the first character of <code>str</code> to append. |
| <code>__n</code>   | The number of characters to append.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                             |
|--------------------------------|---------------------------------------------|
| <code>std::out_of_range</code> | if <code>__pos</code> is not a valid index. |
|--------------------------------|---------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

**append()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::append (
    initializer_list<_CharT> __l) [inline], [constexpr], [inherited]
```

Append an `initializer_list` of characters.

**Parameters**

|                |                                                            |
|----------------|------------------------------------------------------------|
| <code>↵</code> | The <code>initializer_list</code> of characters to append. |
| <code>↵</code> |                                                            |
| <code>↵</code> |                                                            |
| <code>↵</code> |                                                            |
| <code>/</code> |                                                            |

**Returns**

Reference to this string.

**append()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
    size_type __n,
    _CharT __c) [inline], [constexpr], [inherited]
```

Append multiple characters.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__n</code> | The number of characters to append. |
| <code>__c</code> | The character to use.               |

**Returns**

Reference to this string.

Appends `__n` copies of `__c` to this string.

**assign()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [constexpr], [noexcept],
[inherited]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

**assign()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const _CharT * __s,
    size_type __n) [inline], [constexpr], [inherited]
```

Set value to a C substring.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | The C string to use.         |
| <code>__n</code> | Number of characters to use. |

**Returns**

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

**assign()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr], [inherited]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Referenced by `std::basic_string< _CharT >::assign()`, `std::basic_string< _CharT >::assign()`, `std::basic_string< _CharT >::assign()`, `__gnu_debug::basic_string< char >::compare()`, `std::basic_string< _CharT >::operator=()`, `std::basic_string< _CharT >::operator=()`, `std::basic_string< _CharT >::operator=()`, `std::basic_string< _CharT >::operator=()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`

**assign()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n = npos) [inline], [constexpr], [inherited]
```

Set value to a substring of a string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

**assign()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    size_type __n,
    _CharT __c) [inline], [constexpr], [inherited]
```

Set value to multiple characters.

**Parameters**

|                  |                                 |
|------------------|---------------------------------|
| <code>__n</code> | Length of the resulting string. |
| <code>__c</code> | The character to use.           |

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

**at()** [1/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::at (
    size_type __n) [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                                        |
|--------------------------------|----------------------------------------|
| <code>std::out_of_range</code> | If <code>n</code> is an invalid index. |
|--------------------------------|----------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [2/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::at (
    size_type __n) [inline]
```

Provides access to the data contained in the string.



**Parameters**

|                                      |                                       |
|--------------------------------------|---------------------------------------|
| $\leftrightarrow$<br><code>_n</code> | The index of the character to access. |
|--------------------------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                                  |
|--------------------------------|----------------------------------|
| <code>std::out_of_range</code> | If <i>n</i> is an invalid index. |
|--------------------------------|----------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

**at()** [3/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::at (
    size_type __n) const [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

**Parameters**

|                                      |                                       |
|--------------------------------------|---------------------------------------|
| $\leftrightarrow$<br><code>_n</code> | The index of the character to access. |
|--------------------------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                |                                  |
|--------------------------------|----------------------------------|
| <code>std::out_of_range</code> | If <i>n</i> is an invalid index. |
|--------------------------------|----------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [4/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::at (
    size_type __n) const [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                                      |                                       |
|--------------------------------------|---------------------------------------|
| $\leftrightarrow$<br><code>_n</code> | The index of the character to access. |
|--------------------------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If $n$ is an invalid index. |
|--------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::at (
    size_type __n) [inline], [nodiscard], [constexpr], [inherited]
```

Provides access to the data contained in the string.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If $n$ is an invalid index. |
|--------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::at (
    size_type __n) const [inline], [nodiscard], [constexpr], [inherited]
```

Provides access to the data contained in the string.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

## Exceptions

|                                |                                  |
|--------------------------------|----------------------------------|
| <code>std::out_of_range</code> | If <i>n</i> is an invalid index. |
|--------------------------------|----------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

### **back()** [1/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::back () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

### **back()** [2/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::back () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the last element of the string.

### **capacity()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::capacity () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

### **compare()** [1/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
    const basic_string< _CharT, _Traits, _Allocator > & __str) const [inline], [nodiscard],
[constexpr]
```

Compare to a string.

## Parameters

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

## Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [2/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
    size_type __pos,
    size_type __n,
    const basic_string<_CharT, _Traits, _Allocator> & __str) const [inline], [nodiscard],
[constexpr]
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [3/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
    size_type __pos,
    size_type __n,
    const basic_string<_CharT, _Traits, _Allocator> & __str) const [inline]
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [4/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s) const [inline], [nodiscard], [constexpr]
```

Compare substring to a C string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [5/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2) const [inline], [nodiscard], [constexpr]
```

Compare substring against a character array.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | character array to compare against.    |
| <code>__n2</code>  | Number of characters of s.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

**compare()** [6/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2) const [inline]
```

Compare substring against a character array.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | character array to compare against.    |
| <code>__n2</code>  | Number of characters of s.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

**compare()** [7/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
    size_type __pos1,
    size_type __n1,
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos2,
    size_type __n2 = npos) const [inline], [nodiscard], [constexpr]
```

Compare substring to a substring.

**Parameters**

|                     |                                               |
|---------------------|-----------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.        |
| <code>__n1</code>   | Number of characters in substring.            |
| <code>__str</code>  | String to compare against.                    |
| <code>__pos2</code> | Index of first character of substring of str. |
| <code>__n2</code>   | Number of characters in substring of str.     |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [8/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
    size_type __pos1,
    size_type __n1,
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos2,
    size_type __n2 = npos) const [inline]
```

Compare substring to a substring.

**Parameters**

|                     |                                               |
|---------------------|-----------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.        |
| <code>__n1</code>   | Number of characters in substring.            |
| <code>__str</code>  | String to compare against.                    |
| <code>__pos2</code> | Index of first character of substring of str. |
| <code>__n2</code>   | Number of characters in substring of str.     |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [9/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
    size_type __pos,
    size_type __n,
    const basic_string<_CharT, _Traits, _Alloc> & __str) const [inline], [nodiscard],
[constexpr], [inherited]
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
|--------------------|----------------------------------------|

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of characters in substring. |
| <code>__str</code> | String to compare against.         |

## Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [10/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s) const [inline], [nodiscard], [constexpr], [inherited]
```

Compare substring to a C string.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

## Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer  $< 0$  if the substring is ordered before `__s`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [11/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2) const [inline], [nodiscard], [constexpr], [inherited]
```

Compare substring against a character array.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |



|                   |                                          |
|-------------------|------------------------------------------|
| <code>__s</code>  | character array to compare against.      |
| <code>__n2</code> | Number of characters of <code>s</code> . |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer  $< 0$  if this substring is ordered before the string from `__s`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

**compare()** [12/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    size_type __pos1,
    size_type __n1,
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos2,
    size_type __n2 = npos) const [inline], [nodiscard], [constexpr], [inherited]
```

Compare substring to a substring.

**Parameters**

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.                      |
| <code>__n1</code>   | Number of characters in substring.                          |
| <code>__str</code>  | String to compare against.                                  |
| <code>__pos2</code> | Index of first character of substring of <code>str</code> . |
| <code>__n2</code>   | Number of characters in substring of <code>str</code> .     |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**copy()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::copy (
    _CharT * __s,
    size_type __n,
    size_type __pos = 0) const [constexpr], [inherited]
```

Copy substring into C string.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

## Returns

Number of characters actually copied

## Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

**data()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string<_CharT, _Traits, _Alloc>::data () [inline], [nodiscard], [constexpr],
[noexcept], [inherited]
```

Return non-const pointer to contents.

This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

**empty()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
bool std::basic_string<_CharT, _Traits, _Allocator>::empty () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

**erase()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (
    __const_iterator __first,
    __const_iterator __last) [inline], [constexpr], [inherited]
```

Remove a range of characters.

## Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

## Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
    __const_iterator __position) [inline], [constexpr], [inherited]
```

Remove one character.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

**Returns**

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::iterator std::basic_string< _CharT, _Traits, _Alloc >↵
::erase (
    iterator __first,
    iterator __last) [inherited]
```

Remove a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
    iterator __position) [inline], [inherited]
```

Remove one character.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

**Returns**

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::erase (
    size_type __pos = 0,
    size_type __n = npos) [inline], [constexpr], [inherited]
```

Remove characters.

**Parameters**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
|--------------------------------|-------------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by `std::getline()`, `std::operator>>()`, `std::basic_string<_CharT>::pop_back()`, and `resize()`.

**find()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find (
    _CharT __c,
    size_type __pos = 0) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find (
    _CharT __c,
    size_type __pos = 0) const [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find() [3/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find() [4/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find() [8/11]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
    _CharT __c,
    size_type __pos = 0) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find() [9/11]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [std::data\(\)](#), [npos](#), and [std::size\(\)](#).

Referenced by [std::basic\\_string< \\_CharT >::find\(\)](#), [std::basic\\_string< \\_CharT >::find\(\)](#), [std::basic\\_string< \\_CharT >::find\(\)](#), and [std::basic\\_string< \\_CharT >::find\\_first\\_of\(\)](#).

**find()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find\_first\_not\_of()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (
    _CharT __c,
    size_type __pos = 0) const [nodiscard], [constexpr], [noexcept]
```

Find position of a different character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |



**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of() [2/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
    _CharT __c,
    size_type __pos = 0) const [noexcept]
```

Find position of a different character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of() [3/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of() [7/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
    const basic_string< _CharT, _Traits, _Allocator > & __str,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of() [8/11]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
    _CharT __c,
    size_type __pos = 0) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a different character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_first\_not\_of()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_first\_not\_of()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | String containing characters to avoid. |
|--------------------|----------------------------------------|

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__pos</code> | Index of character to search from (default 0). |
|--------------------|------------------------------------------------|

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_first\\_not\\_of\(\)](#), [std::basic\\_string<\\_CharT>::find\\_first\\_not\\_of\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_first\\_not\\_of\(\)](#).

#### **find\_first\_of()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of (
    _CharT __c,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

#### **find\_first\_of()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of (
    _CharT __c,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

**find\_first\_of()** [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic\_string<_CharT, _Traits, _Allocator>::find_first_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | String containing characters to locate.    |
| <code>__pos</code> | Index of character to search from.         |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic\_string<_CharT, _Traits, _Allocator>::find_first_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | String containing characters to locate.    |
| <code>__pos</code> | Index of character to search from.         |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic\_string<_CharT, _Traits, _Allocator>::find_first_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
    const basic_string< _CharT, _Traits, _Allocator > & __str,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
    const basic_string< _CharT, _Traits, _Allocator > & __str,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic\_string<_CharT, _Traits, _Alloc>::find_first_of (
    _CharT __c,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

**find\_first\_of()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic\_string<_CharT, _Traits, _Alloc>::size_type std::basic\_string<_CharT, _Traits, _Alloc>::find_first_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.                 |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_first\_of()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic\_string<_CharT, _Traits, _Alloc>::find_first_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of C string.



**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string< \\_CharT >::find\\_first\\_of\(\)](#), and [std::basic\\_string< \\_CharT >::find\\_first\\_of\(\)](#).

**find\_last\_not\_of()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
    _CharT __c,
    size_type __pos = npos) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a different character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
    _CharT __c,
    size_type __pos = npos) const [noexcept]
```

Find last position of a different character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find last position of a character not in C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
    const basic_string< _CharT, _Traits, _Allocator > & __str,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
    const basic_string< _CharT, _Traits, _Allocator > & __str,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
    _CharT __c,
    size_type __pos = npos) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a different character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_last\_not\_of()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_last\_not\_of()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string< \\_CharT >::find\\_last\\_not\\_of\(\)](#), [std::basic\\_string< \\_CharT >::find\\_last\\_not\\_of\(\)](#), and [std::basic\\_string< \\_CharT >::find\\_last\\_not\\_of\(\)](#).

**find\_last\_of()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic\_string<_CharT, _Traits, _Allocator>::find_last_of (
    _CharT __c,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

**find\_last\_of()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic\_string<_CharT, _Traits, _Allocator>::find_last_of (
    _CharT __c,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

**find\_last\_of()** [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic\_string<_CharT, _Traits, _Allocator>::find_last_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
    _CharT __c,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |



**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

**find\_last\_of()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_last\_of()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by `std::basic_string<_CharT>::find_last_of()`, and `std::basic_string<_CharT>::find_last_of()`.

**front()** [1/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string<_CharT, _Traits, _Allocator>::front () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

**front()** [2/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string<_CharT, _Traits, _Allocator>::front () [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

**get\_allocator()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
allocator_type std::basic_string<_CharT, _Traits, _Allocator>::get_allocator () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Return copy of allocator used to construct this string.

**insert()** [1/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
    __const_iterator __p,
    _CharT __c) [inline], [constexpr], [inherited]
```

Insert one character.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [2/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
    const_iterator __p,
    _InputIterator __beg,
    _InputIterator __end) [inline], [constexpr], [inherited]
```

Insert a range of characters.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__p</code>   | Const_iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                             |
| <code>__end</code> | End of range.                                               |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [3/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
    const_iterator __p,
    initializer_list< _CharT > __l) [inline], [constexpr], [inherited]
```

Insert an `initializer_list` of characters.

## Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__l</code> | The initializer_list of characters to insert.         |

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

**insert()** [4/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
    const_iterator __p,
    size_type __n,
    _CharT __c) [inline], [constexpr], [inherited]
```

Insert multiple characters.

## Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__p</code> | Const_iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                              |
| <code>__c</code> | The character to insert.                                    |

## Returns

Iterator referencing the first inserted char.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, and `std::basic_string<_CharT>::insert()`.

**insert()** [5/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
    iterator __p,
    _CharT __c) [inline], [inherited]
```

Insert one character.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [6/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
    iterator __p,
    _InputIterator __beg,
    _InputIterator __end) [inline], [inherited]
```

Insert a range of characters.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__p</code>   | Iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                       |
| <code>__end</code> | End of range.                                         |

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[__beg, __end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [7/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
    iterator __p,
    initializer_list< _CharT > __l) [inline], [inherited]
```

Insert an `initializer_list` of characters.

## Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__l</code> | The initializer_list of characters to insert.         |

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

**insert()** [8/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::insert (
    iterator __p,
    size_type __n,
    _CharT __c) [inline], [inherited]
```

Insert multiple characters.

## Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                        |
| <code>__c</code> | The character to insert.                              |

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [9/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
    size_type __pos,
    const _CharT * __s) [inline], [constexpr], [inherited]
```

Insert a C string.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__pos</code> | Position in string to insert at. |
| <code>__s</code>   | The C string to insert.          |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |

Inserts the first *n* characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [10/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::insert (
    size_type __pos,
    const _CharT * __s,
    size_type __n) [inline], [constexpr], [inherited]
```

Insert a C substring.

**Parameters**

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pos</code> | Position in string to insert at.    |
| <code>__s</code>   | The C string to insert.             |
| <code>__n</code>   | The number of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [11/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
    size_type __pos,
    size_type __n,
    _CharT __c) [inline], [constexpr], [inherited]
```

Insert multiple characters.

## Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

## Returns

Reference to this string.

## Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [12/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
    size_type __pos1,
    const basic_string<_CharT, _Traits, _Alloc> & __str) [inline], [constexpr], [inherited]
```

Insert value of a string.

## Parameters

|                     |                                  |
|---------------------|----------------------------------|
| <code>__pos1</code> | Position in string to insert at. |
| <code>__str</code>  | The string to insert.            |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [13/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
    size_type __pos1,
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos2,
    size_type __n = npos) [inline], [constexpr], [inherited]
```

Insert a substring.



**Parameters**

|                     |                                                    |
|---------------------|----------------------------------------------------|
| <code>__pos1</code> | Position in string to insert at.                   |
| <code>__str</code>  | The string to insert.                              |
| <code>__pos2</code> | Start of characters in <code>str</code> to insert. |
| <code>__n</code>    | Number of characters to insert.                    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                                           |
|--------------------------------|---------------------------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                           |
| <code>std::out_of_range</code> | If <code>pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**length()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::length () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

**max\_size()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::max_size () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the `size()` of the largest possible string.

**operator+=()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator+= (
    const basic_string<_CharT, _Traits, _Alloc> & __str) [inline], [constexpr], [inherited]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

**operator[]()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic\_string<\_CharT, \_Traits, \_Alloc>::operator\[\] (
    size_type __pos) [inline], [nodiscard], [constexpr], [inherited]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**operator[]()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic\_string<\_CharT, \_Traits, \_Alloc>::operator\[\] (
    size_type __pos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Referenced by [std::basic\\_string<\\_CharT>::back\(\)](#), [std::basic\\_string<\\_CharT>::back\(\)](#), [std::basic\\_string<\\_CharT>::back\(\)](#), [std::basic\\_string<\\_CharT>::front\(\)](#), [std::basic\\_string<\\_CharT>::front\(\)](#), and [std::basic\\_string<\\_CharT>::front\(\)](#).

**replace()** [1/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic\_string & std::basic\_string<\_CharT, \_Traits, \_Alloc>::replace (
    __const_iterator __i1,
    __const_iterator __i2,
    const _CharT * __s) [inline], [constexpr], [inherited]
```

Replace range of characters with C string.

**Parameters**

|                                         |                                                 |
|-----------------------------------------|-------------------------------------------------|
| <code><a href="#">_↔</a><br/>_i1</code> | Iterator referencing start of range to replace. |
| <code><a href="#">_↔</a><br/>_i2</code> | Iterator referencing end of range to replace.   |
| <code><a href="#">_↔</a><br/>_s</code>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [2/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    __const_iterator __i1,
    __const_iterator __i2,
    const _CharT * __s,
    size_type __n) [inline], [constexpr], [inherited]
```

Replace range of characters with C substring.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |
| <code>__n</code>  | Number of characters from s to insert.          |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [3/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    __const_iterator __i1,
    __const_iterator __i2,
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr], [inherited]
```

Replace range of characters with string.

## Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [4/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    __const_iterator __i1,
    __const_iterator __i2,
    size_type __n,
    _CharT __c) [inline], [constexpr], [inherited]
```

Replace range of characters with multiple characters.

## Parameters

|                         |                                                 |
|-------------------------|-------------------------------------------------|
| <code>↵<br/>__i1</code> | Iterator referencing start of range to replace. |
| <code>↵<br/>__i2</code> | Iterator referencing end of range to replace.   |
| <code>↵<br/>__n</code>  | Number of characters to insert.                 |
| <code>↵<br/>__c</code>  | Character to insert.                            |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [5/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    const_iterator __i1,
    const_iterator __i2,
    _InputIterator __k1,
    _InputIterator __k2) [inline], [constexpr], [inherited]
```

Replace range of characters with range.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [6/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    const_iterator __i1,
    const_iterator __i2,
    initializer_list< _CharT > __l) [inline], [constexpr], [inherited]
```

Replace range of characters with `initializer_list`.

**Parameters**

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.            |
| <code>__i2</code> | Iterator referencing end of range to replace.              |
| <code>__l</code>  | The <code>initializer_list</code> of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [7/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    iterator __i1,
    iterator __i2,
    _InputIterator __k1,
    _InputIterator __k2) [inline], [inherited]
```

Replace range of characters with range.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [8/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    iterator __i1,
    iterator __i2,
    const _CharT * __s) [inline], [inherited]
```

Replace range of characters with C string.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [9/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    iterator __i1,
    iterator __i2,
    const _CharT * __s,
    size_type __n) [inline], [inherited]
```

Replace range of characters with C substring.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |
| <code>__n</code>  | Number of characters from s to insert.          |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [10/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    iterator __i1,
    iterator __i2,
    const basic_string<_CharT, _Traits, _Alloc > & __str) [inline], [inherited]
```

Replace range of characters with string.

## Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [11/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    iterator __i1,
    iterator __i2,
    initializer_list<_CharT> __l) [inline], [inherited]
```

Replace range of characters with `initializer_list`.

## Parameters

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.            |
| <code>__i2</code> | Iterator referencing end of range to replace.              |
| <code>__l</code>  | The <code>initializer_list</code> of characters to insert. |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [12/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    iterator __i1,
```



```

        iterator __i2,
        size_type __n,
        _CharT __c) [inline], [inherited]

```

Replace range of characters with multiple characters.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__n</code>  | Number of characters to insert.                 |
| <code>__c</code>  | Character to insert.                            |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

#### **replace()** [13/17]

```

template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos,
    size_type __n,
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr], [inherited]

```

Replace characters with value from another string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |

Removes the characters in the range `[_pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, and `std::basic_string<_CharT>::replace()`.

### **replace()** [14/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    size_type __pos,
    size_type __n1,
    const _CharT * __s) [inline], [constexpr], [inherited]
```

Replace characters with value of a C string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> .               |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[_pos, __pos + __n1)` from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### **replace()** [15/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::replace
(
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2) [inline], [constexpr], [inherited]
```

Replace characters with value of a C substring.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | Index of first character to replace.             |
| <code>__n1</code>  | Number of characters to be replaced.             |
| <code>__s</code>   | C string to insert.                              |
| <code>__n2</code>  | Number of characters from <code>s</code> to use. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos1 &gt; size()</code> .              |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [16/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    size_type __pos,
    size_type __n1,
    size_type __n2,
    _CharT __c) [inline], [constexpr], [inherited]
```

Replace characters with multiple characters.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> .             |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [17/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos1,
    size_type __n1,
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos2,
    size_type __n2 = npos) [inline], [constexpr], [inherited]
```

Replace characters with value from another string.

**Parameters**

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos1</code> | Index of first character to replace.    |
| <code>__n1</code>   | Number of characters to be replaced.    |
| <code>__str</code>  | String to insert.                       |
| <code>__pos2</code> | Index of first character of str to use. |
| <code>__n2</code>   | Number of characters from str to use.   |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                                               |
|--------------------------------|-------------------------------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                               |

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**reserve()** [1/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve () [constexpr]
Equivalent to shrink_to_fit().
```

**reserve()** [2/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve (
    size_type __res_arg) [constexpr]
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

**Exceptions**

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

**reserve() [3/4]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string<_CharT, _Traits, _Allocator>::reserve (
    size_type __res_arg)
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

**Exceptions**

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

**reserve() [4/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::reserve (
    size_type __res_arg) [constexpr], [inherited]
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

**Exceptions**

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

References [capacity\(\)](#), [get\\_allocator\(\)](#), and [std::size\(\)](#).

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::tr2::operator>>\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#).

### **resize()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::resize (
    size_type __n) [inline], [constexpr], [inherited]
```

Resizes the string to the specified number of characters.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
|------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

### **resize()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::resize (
    size_type __n,
    _CharT __c) [constexpr], [inherited]
```

Resizes the string to the specified number of characters.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
| <code>__c</code> | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

References [append\(\)](#), [erase\(\)](#), and [size\(\)](#).

Referenced by [std::basic\\_string<\\_CharT>::resize\(\)](#).

### **rfind()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (
    _CharT __c,
    size_type __pos = npos) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**rfind()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind (
    _CharT __c,
    size_type __pos = npos) const [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**rfind()** [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find last position of a C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string to locate.                        |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr]
```

Find last position of a C string.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a string.



**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
    const basic_string< _CharT, _Traits, _Allocator > & __str,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
```

```
    _CharT __c,
    size_type __pos = npos) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References `npos`, and `std::size()`.

**rfind()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [std::min\(\)](#), [npos](#), and [std::size\(\)](#).

**rfind()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [inherited]
```

Find last position of a C string.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a string.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__str</code> | String to locate. |
|--------------------|-------------------|

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__pos</code> | Index of character to search back from (default end). |
|--------------------|-------------------------------------------------------|

### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_last\\_of\(\)](#), [std::basic\\_string<\\_CharT>::rfind\(\)](#), [std::basic\\_string<\\_CharT>::rfind\(\)](#), [std::basic\\_string<\\_CharT>::rfind\(\)](#), and [std::basic\\_string<\\_CharT>::rfind\(\)](#).

### size()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::size () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

### substr()

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string<_CharT, _Traits, _Alloc>::substr (
    size_type __pos = 0,
    size_type __n = npos) const [inline], [nodiscard], [constexpr], [inherited]
```

Get a substring.

### Parameters

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

### Returns

The new string.

### Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

### swap()

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::swap (
    basic_string<_CharT, _Traits, _Alloc> & __s) [constexpr], [noexcept], [inherited]
```

Swap contents with another string.

## Parameters

|                 |                      |
|-----------------|----------------------|
| <code>_↔</code> | String to swap with. |
| <code>_s</code> |                      |

Exchanges the contents of this string with that of `__s` in constant time.

References [basic\\_string\(\)](#), and [get\\_allocator\(\)](#).

Referenced by [std::basic\\_string<\\_CharT >::assign\(\)](#), and [std::basic\\_string<\\_CharT >::operator=\(\)](#).

## 5.232.3 Member Data Documentation

## npos

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
const size_type std::basic_string<_CharT, _Traits, _Allocator >::npos [static]
```

Value returned by various member functions when they fail.

The documentation for this class was generated from the following file:

- [debug/string](#)

## 5.233 std::basic\_string&lt;\_CharT, \_Traits, \_Alloc &gt; Class Template Reference

```
#include <string>
```

Inheritance diagram for `std::basic_string<_CharT, _Traits, _Alloc >`:



## Public Types

- typedef `_Char_alloc_type` **allocator\_type**
- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic\_string >` **const\_iterator**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic\_string >` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_CharT_alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `const value_type &` **const\_reference**
- typedef `std::reverse\_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `std::reverse\_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Alloc_traits::difference_type` **difference\_type**
- typedef `_CharT_alloc_traits::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic\_string >` **iterator**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic\_string >` **iterator**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_CharT_alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `value_type &` **reference**
- typedef `std::reverse\_iterator< iterator >` **reverse\_iterator**
- typedef `std::reverse\_iterator< iterator >` **reverse\_iterator**
- typedef `_Alloc_traits::size_type` **size\_type**
- typedef `_CharT_alloc_traits::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- [basic\\_string](#) () noexcept
- constexpr [basic\\_string](#) () noexcept(*/\*conditional\*/*)
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>  
constexpr [basic\\_string](#) (`_InputIterator` \_\_beg, `_InputIterator` \_\_end, const `_Alloc` &\_\_a=`_Alloc`())
- template<class `_InputIterator`>  
[basic\\_string](#) (`_InputIterator` \_\_beg, `_InputIterator` \_\_end, const `_Alloc` &\_\_a=`_Alloc`())
- constexpr [basic\\_string](#) ([basic\\_string](#) &&\_\_str) noexcept
- [basic\\_string](#) ([basic\\_string](#) &&\_\_str) noexcept
- **basic\_string** ([basic\\_string](#) &&\_\_str, const `_Alloc` &\_\_a)
- constexpr **basic\_string** ([basic\\_string](#) &&\_\_str, const `_Alloc` &\_\_a) noexcept(`_Alloc_traits::_S_always_equal`())
- [basic\\_string](#) (const `_Alloc` &\_\_a)
- constexpr [basic\\_string](#) (const `_Alloc` &\_\_a) noexcept
- template<typename = `_RequireAllocator<_Alloc>`>  
constexpr [basic\\_string](#) (const `_CharT` \*\_\_s, const `_Alloc` &\_\_a=`_Alloc`())
- template<typename = `_RequireAllocator<_Alloc>`>  
[basic\\_string](#) (const `_CharT` \*\_\_s, const `_Alloc` &\_\_a=`_Alloc`())
- constexpr [basic\\_string](#) (const `_CharT` \*\_\_s, `size_type` \_\_n, const `_Alloc` &\_\_a=`_Alloc`())
- [basic\\_string](#) (const `_CharT` \*\_\_s, `size_type` \_\_n, const `_Alloc` &\_\_a=`_Alloc`())
- constexpr [basic\\_string](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str)
- constexpr **basic\_string** (const [basic\\_string](#) &\_\_str, const `_Alloc` &\_\_a)

- **basic\_string** (const [basic\\_string](#) &\_\_str, const \_Alloc &\_\_a)
- constexpr [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, const \_Alloc &\_\_a=\_Alloc())
- constexpr [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- constexpr [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n, const \_Alloc &\_\_a)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n, const \_Alloc &\_\_a)
- constexpr [basic\\_string](#) (initializer\_list<\_CharT> \_\_l, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (initializer\_list<\_CharT> \_\_l, const \_Alloc &\_\_a=\_Alloc())
- template<typename = \_RequireAllocator<\_Alloc>>  
constexpr [basic\\_string](#) (size\_type \_\_n, \_CharT \_\_c, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (size\_type \_\_n, \_CharT \_\_c, const \_Alloc &\_\_a=\_Alloc())
- constexpr ~[basic\\_string](#) ()
- ~[basic\\_string](#) () noexcept
- template<typename \_Operation>  
constexpr void [\\_\\_resize\\_and\\_overwrite](#) (size\_type \_\_n, \_Operation \_\_op)
- template<typename \_Operation>  
void [\\_\\_resize\\_and\\_overwrite](#) (size\_type \_\_n, \_Operation \_\_op)
- template<typename \_InputIterator>  
[basic\\_string](#)<\_CharT, \_Traits, \_Alloc> & [\\_M\\_replace\\_dispatch](#) (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2, \_\_false\_type)
- template<typename \_Iterator>  
\_CharT \* [\\_S\\_construct](#) (\_Iterator \_\_beg, \_Iterator \_\_end, const \_Alloc &\_\_a, [forward\\_iterator\\_tag](#))
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr [basic\\_string](#) & [append](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<class \_InputIterator>  
[basic\\_string](#) & [append](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- constexpr [basic\\_string](#) & [append](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [append](#) (const \_CharT \*\_\_s)
- constexpr [basic\\_string](#) & [append](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [append](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- constexpr [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str)
- constexpr [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[npos](#))
- [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[npos](#))
- constexpr [basic\\_string](#) & [append](#) (initializer\_list<\_CharT> \_\_l)
- [basic\\_string](#) & [append](#) (initializer\_list<\_CharT> \_\_l)
- constexpr [basic\\_string](#) & [append](#) (size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & [append](#) (size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr [basic\\_string](#) & [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<class \_InputIterator>  
[basic\\_string](#) & [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- constexpr [basic\\_string](#) & [assign](#) ([basic\\_string](#) &&\_\_str) noexcept(\_Alloc\_traits::\_S\_nothrow\_move())
- [basic\\_string](#) & [assign](#) ([basic\\_string](#) &&\_\_str) noexcept([allocator\\_traits](#)<\_Alloc>::is\_always\_equal::value)
- constexpr [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s)
- constexpr [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- constexpr [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str)

- constexpr `basic_string` & `assign` (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n=`npos`)
- `basic_string` & `assign` (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n=`npos`)
- constexpr `basic_string` & `assign` (initializer\_list< \_CharT > \_\_l)
- `basic_string` & `assign` (initializer\_list< \_CharT > \_\_l)
- constexpr `basic_string` & `assign` (size\_type \_\_n, \_CharT \_\_c)
- `basic_string` & `assign` (size\_type \_\_n, \_CharT \_\_c)
- constexpr reference `at` (size\_type \_\_n)
- reference `at` (size\_type \_\_n)
- constexpr const\_reference `at` (size\_type \_\_n) const
- const\_reference `at` (size\_type \_\_n) const
- reference `back` ()
- constexpr const\_reference `back` () const noexcept
- const\_reference `back` () const noexcept
- constexpr reference `back` () noexcept
- iterator `begin` ()
- constexpr const\_iterator `begin` () const noexcept
- const\_iterator `begin` () const noexcept
- constexpr iterator `begin` () noexcept
- constexpr const\_CharT \* `c_str` () const noexcept
- const\_CharT \* `c_str` () const noexcept
- constexpr size\_type `capacity` () const noexcept
- size\_type `capacity` () const noexcept
- constexpr const\_iterator `cbegin` () const noexcept
- const\_iterator `cbegin` () const noexcept
- constexpr const\_iterator `cend` () const noexcept
- const\_iterator `cend` () const noexcept
- constexpr void `clear` () noexcept
- void `clear` () noexcept
- constexpr int `compare` (const \_CharT \*\_\_s) const noexcept
- int `compare` (const \_CharT \*\_\_s) const noexcept
- constexpr int `compare` (const `basic_string` &\_\_str) const
- int `compare` (const `basic_string` &\_\_str) const
- constexpr int `compare` (size\_type \_\_pos, size\_type \_\_n, const `basic_string` &\_\_str) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n, const `basic_string` &\_\_str) const
- constexpr int `compare` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s) const
- constexpr int `compare` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2) const
- constexpr int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2=`npos`) const
- int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2=`npos`) const
- constexpr size\_type `copy` (\_CharT \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- size\_type `copy` (\_CharT \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- constexpr const\_reverse\_iterator `crbegin` () const noexcept
- const\_reverse\_iterator `crbegin` () const noexcept
- constexpr const\_reverse\_iterator `crend` () const noexcept
- const\_reverse\_iterator `crend` () const noexcept
- constexpr const\_CharT \* `data` () const noexcept
- const\_CharT \* `data` () const noexcept
- constexpr \_CharT \* `data` () noexcept

- `_CharT * data () noexcept(false)`
- `constexpr bool empty () const noexcept`
- `bool empty () const noexcept`
- `iterator end ()`
- `constexpr const_iterator end () const noexcept`
- `const_iterator end () const noexcept`
- `constexpr iterator end () noexcept`
- `constexpr bool ends_with (_CharT __x) const noexcept`
- `bool ends_with (_CharT __x) const noexcept`
- `constexpr bool ends_with (basic_string_view<_CharT, _Traits> __x) const noexcept`
- `bool ends_with (basic_string_view<_CharT, _Traits> __x) const noexcept`
- `constexpr bool ends_with (const _CharT *__x) const noexcept`
- `bool ends_with (const _CharT *__x) const noexcept`
- `constexpr iterator erase (_const_iterator __first, _const_iterator __last)`
- `constexpr iterator erase (_const_iterator __position)`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `constexpr basic_string & erase (size_type __pos=0, size_type __n=npos)`
- `basic_string & erase (size_type __pos=0, size_type __n=npos)`
- `constexpr size_type find (_CharT __c, size_type __pos=0) const noexcept`
- `size_type find (_CharT __c, size_type __pos=0) const noexcept`
- `constexpr size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `constexpr size_type find (const _CharT *__s, size_type __pos=0) const noexcept`
- `size_type find (const _CharT *__s, size_type __pos=0) const noexcept`
- `constexpr size_type find (const basic_string & __str, size_type __pos=0) const noexcept`
- `size_type find (const basic_string & __str, size_type __pos=0) const noexcept`
- `constexpr size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept`
- `size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept`
- `constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept`
- `size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept`
- `constexpr size_type find_first_not_of (const basic_string & __str, size_type __pos=0) const noexcept`
- `size_type find_first_not_of (const basic_string & __str, size_type __pos=0) const noexcept`
- `constexpr size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept`
- `size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept`
- `constexpr size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `constexpr size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept`
- `size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept`
- `constexpr size_type find_first_of (const basic_string & __str, size_type __pos=0) const noexcept`
- `size_type find_first_of (const basic_string & __str, size_type __pos=0) const noexcept`
- `constexpr size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept`
- `size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept`
- `constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept`
- `size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept`
- `constexpr size_type find_last_not_of (const basic_string & __str, size_type __pos=npos) const noexcept`
- `size_type find_last_not_of (const basic_string & __str, size_type __pos=npos) const noexcept`



- constexpr size\_type [find\\_last\\_of](#) (\_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [find\\_last\\_of](#) (\_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- constexpr size\_type [find\\_last\\_of](#) (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type [find\\_last\\_of](#) (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type [find\\_last\\_of](#) (const \_CharT \* \_\_s, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [find\\_last\\_of](#) (const \_CharT \* \_\_s, size\_type \_\_pos=[npos](#)) const noexcept
- constexpr size\_type [find\\_last\\_of](#) (const [basic\\_string](#) & \_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [find\\_last\\_of](#) (const [basic\\_string](#) & \_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- reference [front](#) ()
- constexpr const\_reference [front](#) () const noexcept
- const\_reference [front](#) () const noexcept
- constexpr reference [front](#) () noexcept
- constexpr allocator\_type [get\\_allocator](#) () const noexcept
- allocator\_type [get\\_allocator](#) () const noexcept
- constexpr iterator [insert](#) (\_\_const\_iterator \_\_p, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>>  
constexpr iterator [insert](#) (const\_iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- constexpr iterator [insert](#) (const\_iterator \_\_p, [initializer\\_list](#)<\_CharT> \_\_l)
- constexpr iterator [insert](#) (const\_iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- iterator [insert](#) (iterator \_\_p, \_CharT \_\_c)
- template<class \_InputIterator>  
void [insert](#) (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void [insert](#) (iterator \_\_p, [initializer\\_list](#)<\_CharT> \_\_l)
- void [insert](#) (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s, size\_type \_\_n)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \* \_\_s, size\_type \_\_n)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) & \_\_str)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) & \_\_str, size\_type \_\_pos2, size\_type \_\_n↵  
n=[npos](#))
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) & \_\_str, size\_type \_\_pos2, size\_type \_\_n=[npos](#))
- constexpr size\_type [length](#) () const noexcept
- size\_type [length](#) () const noexcept
- constexpr size\_type [max\\_size](#) () const noexcept
- size\_type [max\\_size](#) () const noexcept
- constexpr [basic\\_string](#) & [operator+=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator+=](#) (\_CharT \_\_c)
- constexpr [basic\\_string](#) & [operator+=](#) (const \_CharT \* \_\_s)
- [basic\\_string](#) & [operator+=](#) (const \_CharT \* \_\_s)
- constexpr [basic\\_string](#) & [operator+=](#) (const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [operator+=](#) (const [basic\\_string](#) & \_\_str)
- constexpr [basic\\_string](#) & [operator+=](#) ([initializer\\_list](#)<\_CharT> \_\_l)
- [basic\\_string](#) & [operator+=](#) ([initializer\\_list](#)<\_CharT> \_\_l)
- constexpr [basic\\_string](#) & [operator=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator=](#) ([basic\\_string](#) && \_\_str) noexcept([/\\*conditional \\*/](#))
- constexpr [basic\\_string](#) & [operator=](#) ([basic\\_string](#) && \_\_str) noexcept(\_Alloc\_traits::\_S\_nothrow\_move())

- constexpr [basic\\_string](#) & [operator=](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [operator=](#) (const \_CharT \*\_\_s)
- constexpr [basic\\_string](#) & [operator=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [operator=](#) (const [basic\\_string](#) &\_\_str)
- constexpr [basic\\_string](#) & [operator=](#) ([initializer\\_list](#)<\_CharT> \_\_l)
- [basic\\_string](#) & [operator=](#) ([initializer\\_list](#)<\_CharT> \_\_l)
- constexpr reference [operator\[\]](#) (size\_type \_\_pos)
- reference [operator\[\]](#) (size\_type \_\_pos)
- constexpr const\_reference [operator\[\]](#) (size\_type \_\_pos) const noexcept
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const noexcept
- void [pop\\_back](#) ()
- constexpr void [pop\\_back](#) () noexcept
- constexpr void [push\\_back](#) (\_CharT \_\_c)
- void [push\\_back](#) (\_CharT \_\_c)
- [reverse\\_iterator](#) [rbegin](#) ()
- constexpr [const\\_reverse\\_iterator](#) [rbegin](#) () const noexcept
- [const\\_reverse\\_iterator](#) [rbegin](#) () const noexcept
- constexpr [reverse\\_iterator](#) [rbegin](#) () noexcept
- [reverse\\_iterator](#) [rend](#) ()
- constexpr [const\\_reverse\\_iterator](#) [rend](#) () const noexcept
- [const\\_reverse\\_iterator](#) [rend](#) () const noexcept
- constexpr [reverse\\_iterator](#) [rend](#) () noexcept
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \*\_\_s)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const [basic\\_string](#) &\_\_str)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>
  - constexpr [basic\\_string](#) & [replace](#) (const\_iterator \_\_i1, const\_iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (const\_iterator \_\_i1, const\_iterator \_\_i2, [initializer\\_list](#)<\_CharT> \_\_l)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- template<class \_InputIterator>
  - [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, [initializer\\_list](#)<\_CharT> \_\_l)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- constexpr [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) &\_\_str)
- constexpr [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- constexpr [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2)

- `basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)`
- `constexpr basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `constexpr basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str, size_type __pos2, size_type __n2=npos)`
- `basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str, size_type __pos2, size_type __n2=npos)`
- `constexpr void reserve ()`
- `void reserve ()`
- `constexpr void reserve (size_type __res_arg)`
- `void reserve (size_type __res_arg)`
- `constexpr void resize (size_type __n)`
- `void resize (size_type __n)`
- `constexpr void resize (size_type __n, _CharT __c)`
- `void resize (size_type __n, _CharT __c)`
- `constexpr size_type rfind (_CharT __c, size_type __pos=npos) const noexcept`
- `size_type rfind (_CharT __c, size_type __pos=npos) const noexcept`
- `constexpr size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept`
- `constexpr size_type rfind (const _CharT *__s, size_type __pos=npos) const`
- `size_type rfind (const _CharT *__s, size_type __pos=npos) const noexcept`
- `constexpr size_type rfind (const basic_string & __str, size_type __pos=npos) const noexcept`
- `size_type rfind (const basic_string & __str, size_type __pos=npos) const noexcept`
- `constexpr void shrink_to_fit () noexcept`
- `void shrink_to_fit () noexcept`
- `constexpr size_type size () const noexcept`
- `size_type size () const noexcept`
- `constexpr bool starts_with (_CharT __x) const noexcept`
- `bool starts_with (_CharT __x) const noexcept`
- `constexpr bool starts_with (basic_string_view< _CharT, _Traits > __x) const noexcept`
- `bool starts_with (basic_string_view< _CharT, _Traits > __x) const noexcept`
- `constexpr bool starts_with (const _CharT *__x) const noexcept`
- `bool starts_with (const _CharT *__x) const noexcept`
- `constexpr basic_string substr (size_type __pos=0, size_type __n=npos) const`
- `basic_string substr (size_type __pos=0, size_type __n=npos) const`
- `constexpr void swap (basic_string & __s) noexcept`
- `void swap (basic_string & __s) noexcept(*conditional *)`

### Static Public Attributes

- static const size\_type `npos`

### Protected Types

- `typedef const_iterator __const_iterator`
- `typedef iterator __const_iterator`

### Friends

- `template<typename, typename, typename>`  
class `basic_stringbuf`

## 5.233.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_string< _CharT, _Traits, _Alloc >
```

Managing sequences of characters and character-like objects.

Since

C++98

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only `push_back`, `at`, and array access are supported.

Since

C++98

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only `push_back`, `at`, and array access are supported.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_5\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_5_style.html)

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

                                     [_Rep]
                                     _M_length
[basic_string<char_type>]           _M_capacity
_M_dataptr                         _M_refcount
_M_p ----->                      unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single `add` instruction: `_Rep::_M_data()`, and `string↔::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a Copy-On-Write C++ string implementation.

### 5.233.2 Constructor & Destructor Documentation

#### basic\_string() [1/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string () [inline], [constexpr], [noexcept]
```

Default constructor creates an empty string.

Referenced by `std::basic_string< _CharT >::assign()`, `std::basic_string< _CharT >::assign()`, `std::basic_string< _CharT >::substr()`, and `swap()`.

#### basic\_string() [2/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const _Alloc & __a) [inline], [explicit], [constexpr], [noexcept]
```

Construct an empty string using allocator *a*.

#### basic\_string() [3/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Construct string with copy of value of *\_\_str*.

##### Parameters

|              |                |
|--------------|----------------|
| <i>__str</i> | Source string. |
|--------------|----------------|

#### basic\_string() [4/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string as copy of a substring.

##### Parameters

|              |                                        |
|--------------|----------------------------------------|
| <i>__str</i> | Source string.                         |
| <i>__pos</i> | Index of first character to copy from. |
| <i>__a</i>   | Allocator to use.                      |

#### basic\_string() [5/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n) [inline], [constexpr]
```

Construct string as copy of a substring.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |

**basic\_string()** [6/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos,
    size_type __n,
    const _Alloc & __a) [inline], [constexpr]
```

Construct string as copy of a substring.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |
| <code>__a</code>   | Allocator to use.                      |

**basic\_string()** [7/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (
    const _CharT * __s,
    size_type __n,
    const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string initialized by a character array.

## Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source character array.                          |
| <code>__n</code> | Number of characters to copy.                    |
| <code>__a</code> | Allocator to use (default is default allocator). |

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

**basic\_string()** [8/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (
    const _CharT * __s,
    const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string as copy of a C string.

## Parameters

|                       |                                                  |
|-----------------------|--------------------------------------------------|
| <code>↔<br/>_s</code> | Source C string.                                 |
| <code>↔<br/>_a</code> | Allocator to use (default is default allocator). |

**basic\_string()** [9/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    size_type __n,
    _CharT __c,
    const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string as multiple characters.

## Parameters

|                       |                                                  |
|-----------------------|--------------------------------------------------|
| <code>↔<br/>_n</code> | Number of characters.                            |
| <code>↔<br/>_c</code> | Character to use.                                |
| <code>↔<br/>_a</code> | Allocator to use (default is default allocator). |

**basic\_string()** [10/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [constexpr], [noexcept]
```

Move construct string.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The newly-created string contains the exact contents of `__str`. `__str` is a valid, but unspecified string.

**basic\_string()** [11/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    initializer_list< _CharT > __l,
    const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string from an initializer list.

## Parameters

|                       |                                      |
|-----------------------|--------------------------------------|
| <code>↔<br/>_l</code> | std::initializer_list of characters. |
|-----------------------|--------------------------------------|

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__a</code> | Allocator to use (default is default allocator). |
|------------------|--------------------------------------------------|

**basic\_string()** [12/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    _InputIterator __beg,
    _InputIterator __end,
    const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string as copy of a range.

## Parameters

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__beg</code> | Start of range.                                  |
| <code>__end</code> | End of range.                                    |
| <code>__a</code>   | Allocator to use (default is default allocator). |

**~basic\_string()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string () [inline], [constexpr]
```

Destroy the string instance.

**basic\_string()** [13/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string () [inline], [noexcept]
```

Default constructor creates an empty string.

**basic\_string()** [14/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const _Alloc & __a) [inline], [explicit]
```

Construct an empty string using allocator *a*.

**basic\_string()** [15/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Construct string with copy of value of *str*.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|



**basic\_string()** [16/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    const _Alloc & __a = _Alloc())
```

Construct string as copy of a substring.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__a</code>   | Allocator to use.                      |

**basic\_string()** [17/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n)
```

Construct string as copy of a substring.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |

**basic\_string()** [18/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n,
    const _Alloc & __a)
```

Construct string as copy of a substring.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |
| <code>__a</code>   | Allocator to use.                      |

**basic\_string()** [19/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const _CharT * __s,
    size_type __n,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string initialized by a character array.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source character array.                          |
| <code>__n</code> | Number of characters to copy.                    |
| <code>__a</code> | Allocator to use (default is default allocator). |

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

**basic\_string()** [20/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const _CharT * __s,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a C string.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source C string.                                 |
| <code>__a</code> | Allocator to use (default is default allocator). |

**basic\_string()** [21/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    size_type __n,
    _CharT __c,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string as multiple characters.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__n</code> | Number of characters.                            |
| <code>__c</code> | Character to use.                                |
| <code>__a</code> | Allocator to use (default is default allocator). |

**basic\_string()** [22/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [noexcept]
```

Move construct string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The newly-created string contains the exact contents of `__str`. `__str` is a valid, but unspecified string.

**basic\_string()** [23/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    initializer_list< _CharT > __l,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string from an initializer list.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__l</code> | std::initializer_list of characters.             |
| <code>__a</code> | Allocator to use (default is default allocator). |

**basic\_string()** [24/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    _InputIterator __beg,
    _InputIterator __end,
    const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a range.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__beg</code> | Start of range.                                  |
| <code>__end</code> | End of range.                                    |
| <code>__a</code>   | Allocator to use (default is default allocator). |

**~basic\_string()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string () [inline], [noexcept]
```

Destroy the string instance.

### 5.233.3 Member Function Documentation

#### \_\_resize\_and\_overwrite() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string<_CharT, _Traits, _Alloc>::__resize_and_overwrite (
    size_type __n,
    _Operation __op) [constexpr]
```

Non-standard version of `resize_and_overwrite` for C++11 and above.

#### \_\_resize\_and\_overwrite() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string<_CharT, _Traits, _Alloc>::__resize_and_overwrite (
    size_type __n,
    _Operation __op)
```

Non-standard version of `resize_and_overwrite` for C++11 and above.

#### append() [1/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::append (
    _InputIterator __first,
    _InputIterator __last) [inline], [constexpr]
```

Append a range of characters.

##### Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

##### Returns

Reference to this string.

Appends characters in the range `[__first, __last)` to this string.

#### append() [2/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::append (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Append a range of characters.

##### Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Appends characters in the range [`__first`,`__last`) to this string.

**append()** [3/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
    const _CharT * __s)    [inline], [constexpr]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

**append()** [4/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
    const _CharT * __s)    [inline]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

**append()** [5/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
    const _CharT * __s,
    size_type __n)    [inline], [constexpr]
```

Append a C substring.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__s</code> | The C string to append.             |
| <code>__n</code> | The number of characters to append. |

**Returns**

Reference to this string.

**append()** [6/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::append (
    const _CharT * __s,
    size_type __n)
```

Append a C substring.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__s</code> | The C string to append.             |
| <code>__n</code> | The number of characters to append. |

**Returns**

Reference to this string.

**append()** [7/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc > & std::basic_string<_CharT, _Traits, _Alloc >::append (
    const basic_string<_CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Referenced by [std::basic\\_string<\\_CharT >::append\(\)](#), [std::basic\\_string<\\_CharT >::append\(\)](#), [std::basic\\_string<\\_CharT >::append\(\)](#), [std::basic\\_string<\\_CharT >::append\(\)](#), [std::\\_\\_cxx11::collate<\\_CharT >::do\\_transform\(\)](#), [std::basic\\_string<\\_CharT >::operator+=\(\(\)\)](#), [std::basic\\_string<\\_CharT >::operator+=\(\(\)\)](#), [std::basic\\_string<\\_CharT >::operator+=\(\(\)\)](#), [std::operator>>\(\)](#), and [resize\(\)](#).

**append()** [8/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::append (
    const basic_string<_CharT, _Traits, _Alloc > & __str)
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

**append()** [9/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n = npos) [inline], [constexpr]
```

Append a substring.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__str</code> | The string to append.                                       |
| <code>__pos</code> | Index of the first character of <code>str</code> to append. |
| <code>__n</code>   | The number of characters to append.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                             |
|--------------------------------|---------------------------------------------|
| <code>std::out_of_range</code> | if <code>__pos</code> is not a valid index. |
|--------------------------------|---------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

**append()** [10/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n = npos)
```

Append a substring.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__str</code> | The string to append.                                       |
| <code>__pos</code> | Index of the first character of <code>str</code> to append. |
| <code>__n</code>   | The number of characters to append.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                             |
|--------------------------------|---------------------------------------------|
| <code>std::out_of_range</code> | if <code>__pos</code> is not a valid index. |
|--------------------------------|---------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

**append()** [11/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::append (
    initializer_list<_CharT > __l) [inline], [constexpr]
```

Append an initializer\_list of characters.

**Parameters**

|   |                                               |
|---|-----------------------------------------------|
| ↔ | The initializer_list of characters to append. |
| ↔ |                                               |
| ↔ |                                               |
| ↔ |                                               |
| / |                                               |

**Returns**

Reference to this string.

**append()** [12/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::append (
    initializer_list<_CharT > __l) [inline]
```

Append an initializer\_list of characters.

**Parameters**

|   |                                               |
|---|-----------------------------------------------|
| ↔ | The initializer_list of characters to append. |
| ↔ |                                               |
| ↔ |                                               |
| ↔ |                                               |
| / |                                               |

**Returns**

Reference to this string.

**append()** [13/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc > & std::basic_string<_CharT, _Traits, _Alloc >::append (
    size_type __n,
    _CharT __c) [inline], [constexpr]
```

Append multiple characters.

**Parameters**

|         |                                     |
|---------|-------------------------------------|
| ↔<br>_n | The number of characters to append. |
| ↔<br>_c | The character to use.               |



**Returns**

Reference to this string.

Appends `__n` copies of `__c` to this string.

**append()** [14/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
    size_type __n,
    _CharT __c)
```

Append multiple characters.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__n</code> | The number of characters to append. |
| <code>__c</code> | The character to use.               |

**Returns**

Reference to this string.

Appends `__n` copies of `__c` to this string.

**assign()** [1/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last) [inline], [constexpr]
```

Set value to a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Sets value of string to characters in the range [`__first`,`__last`).

**assign()** [2/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Set value to a range of characters.

## Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

## Returns

Reference to this string.

Sets value of string to characters in the range [`__first`,`__last`).

**assign()** [3/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [constexpr], [noexcept]
```

Set value to contents of another string.

## Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

## Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

**assign()** [4/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [noexcept]
```

Set value to contents of another string.

## Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

## Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

**assign()** [5/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const _CharT * __s) [inline], [constexpr]
```

Set value to contents of a C string.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | The C string to use. |
|------------------|----------------------|

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

**assign() [6/16]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const _CharT * __s) [inline]
```

Set value to contents of a C string.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | The C string to use. |
|------------------|----------------------|

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

**assign() [7/16]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const _CharT * __s,
    size_type __n) [inline], [constexpr]
```

Set value to a C substring.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | The C string to use.         |
| <code>__n</code> | Number of characters to use. |

**Returns**

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

**assign()** [8/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const _CharT * __s,
    size_type __n)
```

Set value to a C substring.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | The C string to use.         |
| <code>__n</code> | Number of characters to use. |

**Returns**

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

**assign()** [9/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Referenced by `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::assign()`, `__gnu_debug::basic_string<char>::compare()`, `std::basic_string<_CharT>::operator=()`, `std::basic_string<_CharT>::operator=()`, `std::basic_string<_CharT>::operator=()`, `std::basic_string<_CharT>::operator=()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

**assign()** [10/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

**assign()** [11/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n = npos) [inline], [constexpr]
```

Set value to a substring of a string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

**assign()** [12/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n = npos) [inline]
```

Set value to a substring of a string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

**assign()** [13/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::assign (
    initializer_list<_CharT > __l) [inline], [constexpr]
```

Set value to an initializer\_list of characters.

**Parameters**

|   |                                               |
|---|-----------------------------------------------|
| ↔ | The initializer_list of characters to assign. |
| ↔ |                                               |
| ↔ |                                               |
| ↔ |                                               |
| / |                                               |

**Returns**

Reference to this string.

**assign()** [14/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::assign (
    initializer_list<_CharT > __l) [inline]
```

Set value to an initializer\_list of characters.

**Parameters**

|   |                                               |
|---|-----------------------------------------------|
| ↔ | The initializer_list of characters to assign. |
| ↔ |                                               |
| ↔ |                                               |
| ↔ |                                               |
| / |                                               |

**Returns**

Reference to this string.

**assign()** [15/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::assign (
    size_type __n,
    _CharT __c) [inline], [constexpr]
```

Set value to multiple characters.

**Parameters**

|          |                                 |
|----------|---------------------------------|
| ↔<br>__n | Length of the resulting string. |
| ↔<br>__c | The character to use.           |

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

**assign()** [16/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
    size_type __n,
    _CharT __c) [inline]
```

Set value to multiple characters.

**Parameters**

|                  |                                 |
|------------------|---------------------------------|
| <code>__n</code> | Length of the resulting string. |
| <code>__c</code> | The character to use.           |

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

**at()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
    size_type __n) [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                                        |
|--------------------------------|----------------------------------------|
| <code>std::out_of_range</code> | If <code>n</code> is an invalid index. |
|--------------------------------|----------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
    size_type __n) [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                                    |                                       |
|------------------------------------|---------------------------------------|
| <code>_↔</code><br><code>_n</code> | The index of the character to access. |
|------------------------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                                  |
|--------------------------------|----------------------------------|
| <code>std::out_of_range</code> | If <i>n</i> is an invalid index. |
|--------------------------------|----------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

**at()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::at (
    size_type __n) const [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

**Parameters**

|                                    |                                       |
|------------------------------------|---------------------------------------|
| <code>_↔</code><br><code>_n</code> | The index of the character to access. |
|------------------------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                |                                  |
|--------------------------------|----------------------------------|
| <code>std::out_of_range</code> | If <i>n</i> is an invalid index. |
|--------------------------------|----------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::at (
    size_type __n) const [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                                    |                                       |
|------------------------------------|---------------------------------------|
| <code>_↔</code><br><code>_n</code> | The index of the character to access. |
|------------------------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.



## Exceptions

|                                |                                  |
|--------------------------------|----------------------------------|
| <code>std::out_of_range</code> | If <i>n</i> is an invalid index. |
|--------------------------------|----------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

### **back()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back () [inline]
Returns a read/write reference to the data at the last element of the string.
```

### **back()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back () const [inline], [nodiscard],
[constexpr], [noexcept]
Returns a read-only (constant) reference to the data at the last element of the string.
```

### **back()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back () const [inline], [noexcept]
Returns a read-only (constant) reference to the data at the last element of the string.
```

### **back()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back () [inline], [nodiscard], [constexpr],
[noexcept]
Returns a read/write reference to the data at the last element of the string.
```

### **begin()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () [inline]
Returns a read/write iterator that points to the first character in the string. Unshares the string.
```

### **begin()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const [inline], [nodiscard],
[constexpr], [noexcept]
Returns a read-only (constant) iterator that points to the first character in the string.
```

### **begin()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const [inline], [noexcept]
Returns a read-only (constant) iterator that points to the first character in the string.
```

**begin()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic\_string<\_CharT, \_Traits, \_Alloc>::begin\(\) [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write iterator that points to the first character in the string.

Referenced by [std::basic\\_string<\\_CharT>::crend\(\)](#), [std::basic\\_string<\\_CharT>::erase\(\)](#), [std::basic\\_string<\\_CharT>::erase\(\)](#), [std::basic\\_string<\\_CharT>::insert\(\)](#), [std::basic\\_string<\\_CharT>::insert\(\)](#), [std::basic\\_string<\\_CharT>::insert\(\)](#), [std::regex\\_match\(\)](#), [std::regex\\_match\(\)](#), [std::regex\\_replace\(\)](#), [std::regex\\_replace\(\)](#), [std::regex\\_search\(\)](#), [std::basic\\_string<\\_CharT>::rend\(\)](#), [std::basic\\_string<\\_CharT>::rend\(\)](#), [std::basic\\_string<\\_CharT>::replace\(\)](#), [std::basic\\_string<\\_CharT>::replace\(\)](#), and [std::basic\\_string<\\_CharT>::replace\(\)](#).

**c\_str()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic\_string<\_CharT, \_Traits, \_Alloc>::c\_str\(\) const [inline], [nodiscard],
[constexpr], [noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Referenced by [std::\\_\\_cxx11::collate<\\_CharT>::do\\_compare\(\)](#), [std::money\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::\\_\\_cxx11::collate<\\_CharT>::do\\_transform\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), and [std::regex\\_traits<\\_CharT>::transform\\_primary\(\)](#).

**c\_str()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic\_string<\_CharT, \_Traits, \_Alloc>::c\_str\(\) const [inline], [noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

**capacity()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic\_string<\_CharT, \_Traits, \_Alloc>::capacity\(\) const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Referenced by [std::basic\\_string<\\_CharT>::assign\(\)](#), [std::basic\\_string<\\_CharT>::push\\_back\(\)](#), [reserve\(\)](#), and [reserve\(\)](#).

**capacity()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic\_string<\_CharT, \_Traits, \_Alloc>::capacity\(\) const [inline], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

**cbegin()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic\_string<\_CharT, \_Traits, \_Alloc>::cbegin\(\) const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**cbegin()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**cend()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

**cend()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

**clear()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear () [inline], [constexpr], [noexcept]
```

Erases the string, making it empty.  
Referenced by [std::\\_\\_cxx11::collate< \\_CharT >::do\\_transform\(\)](#).

**clear()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear () [inline], [noexcept]
```

Erases the string, making it empty.

**compare()** [1/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    const _CharT * __s) const [inline], [nodiscard], [constexpr], [noexcept]
```

Compare to a C string.

**Parameters**

|                                                                                                                      |                              |
|----------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a> <code>__s</code> | C string to compare against. |
|----------------------------------------------------------------------------------------------------------------------|------------------------------|

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__s`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [2/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    const _CharT * __s) const [inline], [noexcept]
```

Compare to a C string.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | C string to compare against. |
|------------------|------------------------------|

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [3/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    const basic_string< _CharT, _Traits, _Alloc > & __str) const [inline], [nodiscard],
[constexpr]
```

Compare to a string.

**Parameters**

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [4/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    const basic_string< _CharT, _Traits, _Alloc > & __str) const [inline]
```

Compare to a string.

**Parameters**

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [5/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    size_type __pos,
    size_type __n,
    const basic_string< _CharT, _Traits, _Alloc > & __str) const [inline], [nodiscard],
[constexpr]
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [6/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    size_type __pos,
    size_type __n,
    const basic_string< _CharT, _Traits, _Alloc > & __str) const [inline]
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [7/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s) const [inline], [nodiscard], [constexpr]
```

Compare substring to a C string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [8/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s) const [inline]
```

Compare substring to a C string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [9/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2) const [inline], [nodiscard], [constexpr]
```

Compare substring against a character array.

**Parameters**

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__pos</code> | Index of first character of substring.   |
| <code>__n1</code>  | Number of characters in substring.       |
| <code>__s</code>   | character array to compare against.      |
| <code>__n2</code>  | Number of characters of <code>s</code> . |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

**compare()** [10/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2) const [inline]
```

Compare substring against a character array.

**Parameters**

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__pos</code> | Index of first character of substring.   |
| <code>__n1</code>  | Number of characters in substring.       |
| <code>__s</code>   | character array to compare against.      |
| <code>__n2</code>  | Number of characters of <code>s</code> . |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

**compare()** [11/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
    size_type __pos1,
    size_type __n1,
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos2,
    size_type __n2 = npos) const [inline], [nodiscard], [constexpr]
```

Compare substring to a substring.

**Parameters**

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.                      |
| <code>__n1</code>   | Number of characters in substring.                          |
| <code>__str</code>  | String to compare against.                                  |
| <code>__pos2</code> | Index of first character of substring of <code>str</code> . |
| <code>__n2</code>   | Number of characters in substring of <code>str</code> .     |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [12/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
    size_type __pos1,
    size_type __n1,
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos2,
    size_type __n2 = npos) const [inline]
```

Compare substring to a substring.



**Parameters**

|                     |                                               |
|---------------------|-----------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.        |
| <code>__n1</code>   | Number of characters in substring.            |
| <code>__str</code>  | String to compare against.                    |
| <code>__pos2</code> | Index of first character of substring of str. |
| <code>__n2</code>   | Number of characters in substring of str.     |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**copy() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::copy (
    _CharT * __s,
    size_type __n,
    size_type __pos = 0) const [constexpr]
```

Copy substring into C string.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

**Returns**

Number of characters actually copied

**Exceptions**

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

**copy() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::copy (
    _CharT * __s,
    size_type __n,
    size_type __pos = 0) const
```

Copy substring into C string.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

## Returns

Number of characters actually copied

## Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

**crbegin()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crbegin () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**crbegin()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**crend()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crend () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

**crend()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

**data()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string<_CharT, _Traits, _Alloc>::data () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Return const pointer to contents.

This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).

Referenced by `std::basic_regex<char>::assign()`, `std::basic_string<_CharT>::compare()`, `std::basic_string<_CharT>::compare()`, `std::basic_string<_CharT>::compare()`, `std::__cxx11::collate<_CharT>::do_compare()`, `std::__cxx11::collate<_CharT>::do_transform()`, `std::basic_string<_CharT>::find()`, `std::basic_string<_CharT>::find_first_not_of()`, `std::basic_string<_CharT>::find_first_of()`, `std::basic_string<_CharT>::find_last_of()`, `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>>::from_bytes()`, `std::operator==()`, `std::wstring_convert<Codecvt, _Elem, _Wide_alloc, _Byte_alloc>::from_bytes()`, and `std::regex_traits<_CharT>::transform()`.

#### **data()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string<_CharT, _Traits, _Alloc>::data () const [inline], [noexcept]
```

Return const pointer to contents.

This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).

#### **data()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string<_CharT, _Traits, _Alloc>::data () [inline], [nodiscard], [constexpr],
[noexcept]
```

Return non-const pointer to contents.

This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

#### **data()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string<_CharT, _Traits, _Alloc>::data () [inline]
```

Return non-const pointer to contents.

This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

The standard requires this function to be `noexcept` but for the Copy-On-Write string implementation it can throw. This function allows modifying the string contents directly, which means we must copy-on-write to unshare it, which requires allocating memory.

#### **empty()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string<_CharT, _Traits, _Alloc>::empty () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

Referenced by `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::front()`, `std::basic_string<_CharT>::front()`, `std::basic_string<_CharT>::front()`, `std::tr2::operator>>()`, `std::basic_string<_CharT>::pop_back()`, and `std::basic_string<_CharT>::pop_back()`.

#### **empty()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string<_CharT, _Traits, _Alloc>::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

**end()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::end () [inline]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

**end()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::end () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

**end()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

**end()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::end () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write iterator that points one past the last character in the string.

Referenced by [std::basic\\_string<\\_CharT>::append\(\)](#), [std::basic\\_string<\\_CharT>::crbegin\(\)](#), [std::basic\\_string<\\_CharT>::erase\(\)](#), [std::basic\\_string<\\_CharT>::erase\(\)](#), [std::basic\\_string<\\_CharT>::insert\(\)](#), [std::basic\\_string<\\_CharT>::insert\(\)](#), [std::basic\\_string<\\_CharT>::insert\(\)](#), [std::basic\\_string<\\_CharT>::rbegin\(\)](#), [std::basic\\_string<\\_CharT>::rbegin\(\)](#), [std::basic\\_string<\\_CharT>::rbegin\(\)](#), [std::regex\\_match\(\)](#), [std::regex\\_match\(\)](#), [std::regex\\_replace\(\)](#), [std::regex\\_replace\(\)](#), [std::regex\\_search\(\)](#), [std::basic\\_string<\\_CharT>::replace\(\)](#), [std::basic\\_string<\\_CharT>::replace\(\)](#), and [std::basic\\_string<\\_CharT>::](#)

**erase()** [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (
    __const_iterator __first,
    __const_iterator __last) [inline], [constexpr]
```

Remove a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

**erase()** [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (
    __const_iterator __position) [inline], [constexpr]
```

Remove one character.

## Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

## Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (
    iterator __first,
    iterator __last)
```

Remove a range of characters.

## Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

## Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (
    iterator __position) [inline]
```

Remove one character.

## Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

## Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::erase (
    size_type __pos = 0,
    size_type __n = npos) [inline], [constexpr]
```

Remove characters.

**Parameters**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
|--------------------------------|-------------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by [std::getline\(\)](#), [std::operator>>\(\)](#), [std::basic\\_string<\\_CharT>::pop\\_back\(\)](#), and [resize\(\)](#).

**erase() [6/6]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::erase (
    size_type __pos = 0,
    size_type __n = npos) [inline]
```

Remove characters.

**Parameters**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
|--------------------------------|-------------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**find() [1/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    _CharT __c,
    size_type __pos = 0) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    _CharT __c,
    size_type __pos = 0) const [noexcept]
```

Find position of a character.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a C substring.

## Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |



**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [std::data\(\)](#), [npos](#), and [std::size\(\)](#).

Referenced by [std::basic\\_string<\\_CharT>::find\(\)](#), [std::basic\\_string<\\_CharT>::find\(\)](#), [std::basic\\_string<\\_CharT>::find\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_first\\_of\(\)](#).

**find()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find\_first\_not\_of()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
    _CharT __c,
    size_type __pos = 0) const [nodiscard], [constexpr], [noexcept]
```

Find position of a different character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_first\_not\_of()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
    _CharT __c,
    size_type __pos = 0) const [noexcept]
```

Find position of a different character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C substring.

## Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_first\_not\_of()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find position of a character not in C substring.

## Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C string.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_first\\_not\\_of\(\)](#), [std::basic\\_string<\\_CharT>::find\\_first\\_not\\_of\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_first\\_not\\_of\(\)](#).

**find\_first\_not\_of()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
    _CharT __c,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

**find\_first\_of()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
    _CharT __c,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

**find\_first\_of()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | String containing characters to locate.    |
| <code>__pos</code> | Index of character to search from.         |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_first\_of()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | String containing characters to locate.    |
| <code>__pos</code> | Index of character to search from.         |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic\_string<\_CharT, \_Traits, \_Alloc>::find\_first\_of (
    const _CharT * __s,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic\_string<\_CharT, \_Traits, \_Alloc>::find\_first\_of (
    const basic\_string<\_CharT, \_Traits, \_Alloc> & __str,
    size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_first\\_of\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_first\\_of\(\)](#).

**find\_first\_of()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic\_string<\_CharT, \_Traits, \_Alloc>::find\_first\_of (
    const basic\_string<\_CharT, \_Traits, \_Alloc> & __str,
    size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |



**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of() [1/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
    _CharT __c,
    size_type __pos = npos) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a different character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_last\_not\_of() [2/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
    _CharT __c,
    size_type __pos = npos) const [noexcept]
```

Find last position of a different character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of() [3/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C substring.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_last\_not\_of()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find last position of a character not in C substring.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C string.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_last\\_not\\_of\(\)](#), [std::basic\\_string<\\_CharT>::find\\_last\\_not\\_of\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_last\\_not\\_of\(\)](#).

**find\_last\_not\_of()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
    _CharT __c,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

**find\_last\_of()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
    _CharT __c,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

**find\_last\_of()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.  |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_last\_of()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.  |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_last\\_of\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_last\\_of\(\)](#).

**find\_last\_of()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**front()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic\_string< _CharT, _Traits, _Alloc >::front () [inline]
```

Returns a read/write reference to the data at the first element of the string.

**front()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic\_string< _CharT, _Traits, _Alloc >::front () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

**front()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic\_string< _CharT, _Traits, _Alloc >::front () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

**front()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic\_string< _CharT, _Traits, _Alloc >::front () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the first element of the string.

**get\_allocator()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic\_string< _CharT, _Traits, _Alloc >::get_allocator () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Return copy of allocator used to construct this string.

Referenced by [std::basic\\_string< \\_CharT >::~basic\\_string\(\)](#), [std::basic\\_string< \\_CharT >::assign\(\)](#), [std::basic\\_string< \\_CharT >::clear\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [reserve\(\)](#), [reserve\(\)](#), and [swap\(\)](#).

**get\_allocator()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic\_string< _CharT, _Traits, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Return copy of allocator used to construct this string.

**insert()** [1/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic\_string< _CharT, _Traits, _Alloc >::insert (
    __const_iterator __p,
    _CharT __c) [inline], [constexpr]
```

Insert one character.

## Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

## Returns

Iterator referencing newly inserted char.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [2/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
    const_iterator __p,
    _InputIterator __beg,
    _InputIterator __end) [inline], [constexpr]
```

Insert a range of characters.

## Parameters

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__p</code>   | Const_iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                             |
| <code>__end</code> | End of range.                                               |

## Returns

Iterator referencing the first inserted char.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [3/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
    const_iterator __p,
    initializer_list<_CharT> __l) [inline], [constexpr]
```

Insert an initializer\_list of characters.



**Parameters**

|                                                                                                                       |                                                       |
|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| <a href="#"></a><br><code>__p</code> | Iterator referencing location in string to insert at. |
| <a href="#"></a><br><code>__l</code> | The initializer_list of characters to insert.         |

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

**insert() [4/18]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
    const_iterator __p,
    size_type __n,
    _CharT __c) [inline], [constexpr]
```

Insert multiple characters.

**Parameters**

|                                                                                                                         |                                                             |
|-------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| <a href="#"></a><br><code>__p</code>   | Const_iterator referencing location in string to insert at. |
| <a href="#"></a><br><code>__n</code> | Number of characters to insert                              |
| <a href="#"></a><br><code>__c</code> | The character to insert.                                    |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by [std::basic\\_string< \\_CharT >::insert\(\)](#), [std::basic\\_string< \\_CharT >::insert\(\)](#), [std::basic\\_string< \\_CharT >::insert\(\)](#), [std::basic\\_string< \\_CharT >::insert\(\)](#), and [std::basic\\_string< \\_CharT >::insert\(\)](#).

**insert() [5/18]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
    iterator __p,
    _CharT __c) [inline]
```

Insert one character.

## Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

## Returns

Iterator referencing newly inserted char.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [6/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
    iterator __p,
    _InputIterator __beg,
    _InputIterator __end) [inline]
```

Insert a range of characters.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__p</code>   | Iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                       |
| <code>__end</code> | End of range.                                         |

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[__beg, __end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [7/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
    iterator __p,
    initializer_list< _CharT > __l) [inline]
```

Insert an `initializer_list` of characters.

**Parameters**

|                                  |                                                       |
|----------------------------------|-------------------------------------------------------|
| <a href="#"><code>__p</code></a> | Iterator referencing location in string to insert at. |
| <a href="#"><code>__l</code></a> | The initializer_list of characters to insert.         |

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

**insert() [8/18]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc >::insert (
    iterator __p,
    size_type __n,
    _CharT __c) [inline]
```

Insert multiple characters.

**Parameters**

|                                  |                                                       |
|----------------------------------|-------------------------------------------------------|
| <a href="#"><code>__p</code></a> | Iterator referencing location in string to insert at. |
| <a href="#"><code>__n</code></a> | Number of characters to insert                        |
| <a href="#"><code>__c</code></a> | The character to insert.                              |

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert() [9/18]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::insert (
    size_type __pos,
    const _CharT * __s) [inline], [constexpr]
```

Insert a C string.

**Parameters**

|                                    |                                  |
|------------------------------------|----------------------------------|
| <a href="#"><code>__pos</code></a> | Position in string to insert at. |
| <a href="#"><code>__s</code></a>   | The C string to insert.          |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |

Inserts the first  $n$  characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [10/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
    size_type __pos,
    const _CharT * __s) [inline]
```

Insert a C string.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__pos</code> | Position in string to insert at. |
| <code>__s</code>   | The C string to insert.          |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |

Inserts the first  $n$  characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [11/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::insert (
    size_type __pos,
    const _CharT * __s,
    size_type __n) [inline], [constexpr]
```

Insert a C substring.

**Parameters**

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pos</code> | Position in string to insert at.    |
| <code>__s</code>   | The C string to insert.             |
| <code>__n</code>   | The number of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [12/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
    size_type __pos,
    const _CharT * __s,
    size_type __n)
```

Insert a C substring.

**Parameters**

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pos</code> | Position in string to insert at.    |
| <code>__s</code>   | The C string to insert.             |
| <code>__n</code>   | The number of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [13/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
    size_type __pos,
    size_type __n,
    _CharT __c) [inline], [constexpr]
```

Insert multiple characters.

## Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

## Returns

Reference to this string.

## Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [14/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
    size_type __pos,
    size_type __n,
    _CharT __c) [inline]
```

Insert multiple characters.

## Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

## Returns

Reference to this string.

## Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [15/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
    size_type __pos1,
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Insert value of a string.

## Parameters

|                     |                                  |
|---------------------|----------------------------------|
| <code>__pos1</code> | Position in string to insert at. |
| <code>__str</code>  | The string to insert.            |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [16/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
    size_type __pos1,
    const basic_string<_CharT, _Traits, _Alloc> & __str) [inline]
```

Insert value of a string.

## Parameters

|                     |                                  |
|---------------------|----------------------------------|
| <code>__pos1</code> | Position in string to insert at. |
| <code>__str</code>  | The string to insert.            |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [17/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
    size_type __pos1,
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos2,
    size_type __n = npos) [inline], [constexpr]
```

Insert a substring.



**Parameters**

|                     |                                       |
|---------------------|---------------------------------------|
| <code>__pos1</code> | Position in string to insert at.      |
| <code>__str</code>  | The string to insert.                 |
| <code>__pos2</code> | Start of characters in str to insert. |
| <code>__n</code>    | Number of characters to insert.       |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                                           |
|--------------------------------|---------------------------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                           |
| <code>std::out_of_range</code> | If <code>pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [18/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
    size_type __pos1,
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos2,
    size_type __n = npos) [inline]
```

Insert a substring.

**Parameters**

|                     |                                       |
|---------------------|---------------------------------------|
| <code>__pos1</code> | Position in string to insert at.      |
| <code>__str</code>  | The string to insert.                 |
| <code>__pos2</code> | Start of characters in str to insert. |
| <code>__n</code>    | Number of characters to insert.       |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                                           |
|--------------------------------|---------------------------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                           |
| <code>std::out_of_range</code> | If <code>pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**length()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::length() const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Referenced by `std::basic_string<_CharT>::basic_string()`, `std::basic_string<_CharT>::basic_string()`, `std::__cxx11::collate<_CharT>::do_transform()`, `std::regex_replace()`, and `reserve()`.

**length()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::length() const [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

**max\_size()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::max_size() const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the `size()` of the largest possible string.

Referenced by `std::getline()`, and `std::operator>>()`.

**max\_size()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::max_size() const [inline], [noexcept]
```

Returns the `size()` of the largest possible string.

**operator+=()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator+= (
    _CharT __c) [inline], [constexpr]
```

Append a character.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to append. |
|------------------|--------------------------|

**Returns**

Reference to this string.

**operator+=()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator+= (
    _CharT __c) [inline]
```

Append a character.

**Parameters**

|                 |                          |
|-----------------|--------------------------|
| <code>_↔</code> | The character to append. |
| <code>_C</code> |                          |

**Returns**

Reference to this string.

**operator+=() [3/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
    const _CharT * __s) [inline], [constexpr]
```

Append a C string.

**Parameters**

|                 |                         |
|-----------------|-------------------------|
| <code>_↔</code> | The C string to append. |
| <code>_S</code> |                         |

**Returns**

Reference to this string.

**operator+=() [4/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
    const _CharT * __s) [inline]
```

Append a C string.

**Parameters**

|                 |                         |
|-----------------|-------------------------|
| <code>_↔</code> | The C string to append. |
| <code>_S</code> |                         |

**Returns**

Reference to this string.

**operator+=() [5/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

**operator+=() [6/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

**operator+=() [7/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
    initializer_list< _CharT > __l) [inline], [constexpr]
```

Append an initializer\_list of characters.

**Parameters**

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>↵</code>   | The initializer_list of characters to be appended. |
| <code>__↵</code> |                                                    |
| <code>↵</code>   |                                                    |
| <code>__↵</code> |                                                    |
| <code>/</code>   |                                                    |

**Returns**

Reference to this string.

**operator+=() [8/8]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
    initializer_list< _CharT > __l) [inline]
```

Append an initializer\_list of characters.

**Parameters**

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>↵</code>   | The initializer_list of characters to be appended. |
| <code>__↵</code> |                                                    |
| <code>↵</code>   |                                                    |
| <code>__↵</code> |                                                    |
| <code>/</code>   |                                                    |

**Returns**

Reference to this string.

**operator=()** [1/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    _CharT __c) [inline], [constexpr]
```

Set value to string of length 1.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__c</code> | Source character. |
|------------------|-------------------|

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

**operator=()** [2/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    _CharT __c) [inline]
```

Set value to string of length 1.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__c</code> | Source character. |
|------------------|-------------------|

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

**operator=()** [3/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [noexcept]
```

Move assign the value of *str* to this string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

**operator=()** [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [constexpr], [noexcept]
```

Move assign the value of *str* to this string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

**operator=()** [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    const _CharT * __s) [inline], [constexpr]
```

Copy contents of *s* into this string.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__s</code> | Source null-terminated string. |
|------------------|--------------------------------|

**operator=()** [6/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    const _CharT * __s) [inline]
```

Copy contents of *s* into this string.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__s</code> | Source null-terminated string. |
|------------------|--------------------------------|

**operator=()** [7/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Assign the value of *str* to this string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

**operator=()** [8/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Assign the value of *str* to this string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

**operator=()** [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    initializer_list< _CharT > __l) [inline], [constexpr]
```

Set value to string constructed from initializer list.

**Parameters**

|   |                        |
|---|------------------------|
| ↩ | std::initializer_list. |
| ↩ |                        |
| ↩ |                        |
| ↩ |                        |
| ↩ |                        |

**operator=()** [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    initializer_list< _CharT > __l) [inline]
```

Set value to string constructed from initializer list.

**Parameters**

|   |                        |
|---|------------------------|
| ↩ | std::initializer_list. |
| ↩ |                        |
| ↩ |                        |
| ↩ |                        |
| ↩ |                        |

**operator[]()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
    size_type __pos) [inline], [nodiscard], [constexpr]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**operator[]()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
    size_type __pos) [inline]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

**operator[]()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::operator[] (
    size_type __pos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Referenced by `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::front()`, `std::basic_string<_CharT>::front()`, and `std::basic_string<_CharT>::front()`.

**operator[]()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::operator[] (
    size_type __pos) const [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**pop\_back()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::pop_back () [inline]
```

Remove the last character.

The string must be non-empty.

**pop\_back()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::pop_back () [inline], [constexpr], [noexcept]
```

Remove the last character.

The string must be non-empty.



**push\_back()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::push_back (
    _CharT __c) [inline], [constexpr]
```

Append a single character.

**Parameters**

|                         |                      |
|-------------------------|----------------------|
| $\leftrightarrow$<br>_c | Character to append. |
|-------------------------|----------------------|

Referenced by [std::\\_\\_cxx11::collate<\\_CharT>::do\\_transform\(\)](#), [std::basic\\_string<\\_CharT>::operator+=\(\(\)\)](#), [std::tr2::operator>>\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#).

**push\_back()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::push_back (
    _CharT __c) [inline]
```

Append a single character.

**Parameters**

|                         |                      |
|-------------------------|----------------------|
| $\leftrightarrow$<br>_c | Character to append. |
|-------------------------|----------------------|

**rbegin()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin () [inline]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

**rbegin()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**rbegin()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**rbegin()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin () [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

#### rend() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend () [inline]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

#### rend() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

#### rend() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

#### rend() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend () [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

#### replace() [1/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    __const_iterator __i1,
    __const_iterator __i2,
    const _CharT * __s) [inline], [constexpr]
```

Replace range of characters with C string.

#### Parameters

|                                                                                                 |                                                 |
|-------------------------------------------------------------------------------------------------|-------------------------------------------------|
|  <i>__i1</i> | Iterator referencing start of range to replace. |
|  <i>__i2</i> | Iterator referencing end of range to replace.   |
|  <i>__s</i>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [2/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    __const_iterator __i1,
    __const_iterator __i2,
    const _CharT * __s,
    size_type __n) [inline], [constexpr]
```

Replace range of characters with C substring.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |
| <code>__n</code>  | Number of characters from s to insert.          |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [3/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    __const_iterator __i1,
    __const_iterator __i2,
    const basic_string<_CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Replace range of characters with string.

## Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [4/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    __const_iterator __i1,
    __const_iterator __i2,
    size_type __n,
    _CharT __c) [inline], [constexpr]
```

Replace range of characters with multiple characters.

## Parameters

|                         |                                                 |
|-------------------------|-------------------------------------------------|
| <code>↵<br/>__i1</code> | Iterator referencing start of range to replace. |
| <code>↵<br/>__i2</code> | Iterator referencing end of range to replace.   |
| <code>↵<br/>__n</code>  | Number of characters to insert.                 |
| <code>↵<br/>__c</code>  | Character to insert.                            |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [5/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    const_iterator __i1,
    const_iterator __i2,
    _InputIterator __k1,
    _InputIterator __k2) [inline], [constexpr]
```

Replace range of characters with range.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [6/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    const_iterator __i1,
    const_iterator __i2,
    initializer_list< _CharT > __l) [inline], [constexpr]
```

Replace range of characters with `initializer_list`.

**Parameters**

|                         |                                                            |
|-------------------------|------------------------------------------------------------|
| <code>↔<br/>__i1</code> | Iterator referencing start of range to replace.            |
| <code>↔<br/>__i2</code> | Iterator referencing end of range to replace.              |
| <code>↔<br/>__l</code>  | The <code>initializer_list</code> of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>std::length_error</i> | If new length exceeds <code>max_size()</code> . |
|--------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [7/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    iterator __i1,
    iterator __i2,
    _InputIterator __k1,
    _InputIterator __k2) [inline]
```

Replace range of characters with range.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>std::length_error</i> | If new length exceeds <code>max_size()</code> . |
|--------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [8/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    iterator __i1,
    iterator __i2,
    const _CharT * __s) [inline]
```

Replace range of characters with C string.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [9/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    iterator __i1,
    iterator __i2,
    const _CharT * __s,
    size_type __n) [inline]
```

Replace range of characters with C substring.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |
| <code>__n</code>  | Number of characters from s to insert.          |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [10/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    iterator __i1,
    iterator __i2,
    const basic_string<_CharT, _Traits, _Alloc > & __str) [inline]
```

Replace range of characters with string.

## Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [11/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    iterator __i1,
    iterator __i2,
    initializer_list<_CharT> __l) [inline]
```

Replace range of characters with `initializer_list`.

## Parameters

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.            |
| <code>__i2</code> | Iterator referencing end of range to replace.              |
| <code>__l</code>  | The <code>initializer_list</code> of characters to insert. |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [12/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    iterator __i1,
```



```

        iterator __i2,
        size_type __n,
        _CharT __c) [inline]

```

Replace range of characters with multiple characters.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__n</code>  | Number of characters to insert.                 |
| <code>__c</code>  | Character to insert.                            |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

#### **replace()** [13/22]

```

template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
    size_type __pos,
    size_type __n,
    const basic_string<_CharT, _Traits, _Alloc > & __str) [inline], [constexpr]

```

Replace characters with value from another string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, and `std::basic_string<_CharT>::replace()`.

### replace() [14/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    size_type __pos,
    size_type __n,
    const basic_string<_CharT, _Traits, _Alloc> & __str) [inline]
```

Replace characters with value from another string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### replace() [15/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    size_type __pos,
    size_type __n1,
    const _CharT * __s) [inline], [constexpr]
```

Replace characters with value of a C string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> .               |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace() [16/22]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos,
    size_type __n1,
    const _CharT * __s) [inline]
```

Replace characters with value of a C string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> .               |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [17/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::replace
(
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2) [inline], [constexpr]
```

Replace characters with value of a C substring.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | Index of first character to replace.             |
| <code>__n1</code>  | Number of characters to be replaced.             |
| <code>__s</code>   | C string to insert.                              |
| <code>__n2</code>  | Number of characters from <code>s</code> to use. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos1 &gt; size()</code> .              |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [18/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2)
```

Replace characters with value of a C substring.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | Index of first character to replace.             |
| <code>__n1</code>  | Number of characters to be replaced.             |
| <code>__s</code>   | C string to insert.                              |
| <code>__n2</code>  | Number of characters from <code>s</code> to use. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos1 &gt; size()</code> .              |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [19/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos,
    size_type __n1,
    size_type __n2,
    _CharT __c) [inline], [constexpr]
```

Replace characters with multiple characters.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> .             |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [20/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos,
    size_type __n1,
    size_type __n2,
    _CharT __c) [inline]
```

Replace characters with multiple characters.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
|--------------------|--------------------------------------|

## Parameters

|                   |                                      |
|-------------------|--------------------------------------|
| <code>__n1</code> | Number of characters to be replaced. |
| <code>__n2</code> | Number of characters to insert.      |
| <code>__c</code>  | Character to insert.                 |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> .             |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [21/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos1,
    size_type __n1,
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos2,
    size_type __n2 = npos) [inline], [constexpr]
```

Replace characters with value from another string.

## Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos1</code> | Index of first character to replace.    |
| <code>__n1</code>   | Number of characters to be replaced.    |
| <code>__str</code>  | String to insert.                       |
| <code>__pos2</code> | Index of first character of str to use. |
| <code>__n2</code>   | Number of characters from str to use.   |

## Returns

Reference to this string.

## Exceptions

|                                |                                                                               |
|--------------------------------|-------------------------------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                               |

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [22/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos1,
    size_type __n1,
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos2,
    size_type __n2 = npos) [inline]
```

Replace characters with value from another string.

**Parameters**

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos1</code> | Index of first character to replace.    |
| <code>__n1</code>   | Number of characters to be replaced.    |
| <code>__str</code>  | String to insert.                       |
| <code>__pos2</code> | Index of first character of str to use. |
| <code>__n2</code>   | Number of characters from str to use.   |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                                               |
|--------------------------------|-------------------------------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                               |

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**reserve()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve () [constexpr]
```

Equivalent to `shrink_to_fit()`.

References `capacity()`, `get_allocator()`, and `length()`.

Referenced by `std::basic_string< _CharT >::push_back()`, and `std::basic_string< _CharT >::shrink_to_fit()`.

**reserve()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve ()
```

Equivalent to `shrink_to_fit()`.

**reserve()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
    size_type __res_arg) [constexpr]
```

Attempt to preallocate enough memory for specified number of characters.

## Parameters

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

## Exceptions

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

References [capacity\(\)](#), [get\\_allocator\(\)](#), and [std::size\(\)](#).

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::tr2::operator>>\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#).

**reserve()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::reserve (
    size_type __res_arg)
```

Attempt to preallocate enough memory for specified number of characters.

## Parameters

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

## Exceptions

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

**resize()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::resize (
    size_type __n) [inline], [constexpr]
```

Resizes the string to the specified number of characters.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
|------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.



**resize()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
    size_type __n) [inline]
```

Resizes the string to the specified number of characters.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
|------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

**resize()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
    size_type __n,
    _CharT __c) [constexpr]
```

Resizes the string to the specified number of characters.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
| <code>__c</code> | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

References [append\(\)](#), [erase\(\)](#), and [size\(\)](#).

Referenced by [std::basic\\_string< \\_CharT >::resize\(\)](#).

**resize()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
    size_type __n,
    _CharT __c)
```

Resizes the string to the specified number of characters.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
| <code>__c</code> | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

**rfind()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
    _CharT __c,
    size_type __pos = npos) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**rfind()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
    _CharT __c,
    size_type __pos = npos) const [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**rfind()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a C substring.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | C string to locate. |
|------------------|---------------------|

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [std::min\(\)](#), [npos](#), and [std::size\(\)](#).

#### **rfind()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
    const _CharT * __s,
    size_type __pos,
    size_type __n) const [noexcept]
```

Find last position of a C substring.

#### Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

#### **rfind()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr]
```

Find last position of a C string.

#### Parameters

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
    const _CharT * __s,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a C string.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_last\\_of\(\)](#), [std::basic\\_string<\\_CharT>::rfind\(\)](#), [std::basic\\_string<\\_CharT>::rfind\(\)](#), [std::basic\\_string<\\_CharT>::rfind\(\)](#), and [std::basic\\_string<\\_CharT>::rfind\(\)](#).

**rfind()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
    const basic_string<_CharT, _Traits, _Alloc> & __str,
    size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

## shrink\_to\_fit() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic\_string<_CharT, _Traits, _Alloc >::shrink_to_fit () [inline], [constexpr], [noexcept]
A non-binding request to reduce capacity() to size().
```

## shrink\_to\_fit() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic\_string<_CharT, _Traits, _Alloc >::shrink_to_fit () [inline], [noexcept]
A non-binding request to reduce capacity() to size().
```

```
size() [1/2]
```

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::size () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Referenced by `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::append()`, `std::basic_regex<char>::assign()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::at()`, `std::basic_string<_CharT>::at()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::cend()`, `std::basic_string<_CharT>::compare()`, `std::basic_string<_CharT>::compare()`, `std::basic_string<_CharT>::compare()`, `std::basic_string<_CharT>::empty()`, `std::basic_string<_CharT>::end()`, `std::basic_string<_CharT>::end()`, `std::basic_string<_CharT>::end()`, `std::basic_string<_CharT>::find()`, `std::basic_string<_CharT>::find_first_not_of()`, `std::basic_string<_CharT>::find_first_of()`, `__gnu_debug::basic_string<char>::find_last_of()`, `std::basic_string<_CharT>::find_last_of()`, `std::match_results<_BidirectionalIterator>::wstring_convert<_Codecvt, _Elem, _Wide_alloc, _Byte_alloc>::from_bytes()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::length()`, `std::operator+()`, `std::operator+()`, `std::operator+()`, `std::operator+()`, `std::operator+()`, `std::operator==()`, `std::tr2::operator>>()`, `std::basic_string<_CharT>::operator[]()`, `std::basic_string<_CharT>::operator[]()`, `std::basic_string<_CharT>::pop_back()`, `std::basic_string<_CharT>::pop_back()`, `std::basic_string<_CharT>::push_back()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, `resize()`, `std::wstring_convert<_Codecvt, _Elem, _Wide_alloc>::wstring_convert()`, `std::regex_traits<_CharT>::transform()`, and `std::regex_traits<_CharT>::transform_primary()`.

```
size() [2/2]
```

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic\_string< _CharT, _Traits, _Alloc >::size () const [inline], [noexcept]
Returns the number of characters in the string, not including any null-termination.
```

**substr()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
    size_type __pos = 0,
    size_type __n = npos) const [inline], [nodiscard], [constexpr]
```

## Parameters

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

## Returns

The new string.

## Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

**substr()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string<_CharT, _Traits, _Alloc>::substr (
    size_type __pos = 0,
    size_type __n = npos) const [inline]
```

Get a substring.

## Parameters

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

## Returns

The new string.

## Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

**swap()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::swap (
    basic_string<_CharT, _Traits, _Alloc> & __s) [constexpr], [noexcept]
```

Swap contents with another string.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | String to swap with. |
|------------------|----------------------|

Exchanges the contents of this string with that of `__s` in constant time.

References [basic\\_string\(\)](#), and [get\\_allocator\(\)](#).

Referenced by [std::basic\\_string<\\_CharT>::assign\(\)](#), and [std::basic\\_string<\\_CharT>::operator=\(\)](#).

**swap()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
    basic_string< _CharT, _Traits, _Alloc > & __s) [noexcept]
```

Swap contents with another string.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | String to swap with. |
|------------------|----------------------|

Exchanges the contents of this string with that of `__s` in constant time.

**5.233.4 Member Data Documentation****npos**

```
template<typename _CharT, typename _Traits, typename _Alloc>
static const size_type std::basic_string< _CharT, _Traits, _Alloc >::npos [static]
```

Value returned by various member functions when they fail.

Referenced by [find\(\)](#), [find\(\)](#), [find\\_first\\_not\\_of\(\)](#), [find\\_first\\_not\\_of\(\)](#), [find\\_first\\_of\(\)](#), [find\\_last\\_not\\_of\(\)](#), [find\\_last\\_not\\_of\(\)](#), [find\\_last\\_of\(\)](#), [rfind\(\)](#), and [rfind\(\)](#).

The documentation for this class was generated from the following files:

- [basic\\_string.h](#)
- [cow\\_string.h](#)
- [basic\\_string.tcc](#)

**5.234 std::basic\_string\_view< \_CharT, \_Traits > Class Template Reference**

```
#include <>>
```

**Public Types**

- using **const\_iterator**
- using **const\_pointer**
- using **const\_reference**
- using **const\_reverse\_iterator**
- using **difference\_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse\_iterator**
- using **size\_type**
- using **traits\_type**
- using **value\_type**

**Public Member Functions**

- constexpr **basic\_string\_view** (const \_CharT \* \_\_str) noexcept
- constexpr **basic\_string\_view** (const \_CharT \* \_\_str, size\_type \_\_len) noexcept
- constexpr **basic\_string\_view** (const [basic\\_string\\_view](#) &) noexcept=default
- constexpr const\_reference **at** (size\_type \_\_pos) const

- constexpr const\_reference **back** () const noexcept
- constexpr const\_iterator **begin** () const noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr int **compare** ([basic\\_string\\_view](#) \_\_str) const noexcept
- constexpr int **compare** (const \_CharT \* \_\_str) const noexcept
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, [basic\\_string\\_view](#) \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, [basic\\_string\\_view](#) \_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \* \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \* \_\_str, size\_type \_\_n2) const noexcept(false)
- constexpr size\_type **copy** (\_CharT \* \_\_str, size\_type \_\_n, size\_type \_\_pos=0) const
- constexpr [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- constexpr [const\\_reverse\\_iterator](#) **crend** () const noexcept
- constexpr const\_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr const\_reference **front** () const noexcept
- constexpr size\_type **length** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- constexpr [basic\\_string\\_view](#) & **operator=** (const [basic\\_string\\_view](#) &) noexcept=default
- constexpr const\_reference **operator[]** (size\_type \_\_pos) const noexcept
- constexpr [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- constexpr void **remove\_prefix** (size\_type \_\_n) noexcept
- constexpr void **remove\_suffix** (size\_type \_\_n) noexcept
- constexpr [const\\_reverse\\_iterator](#) **rend** () const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **rfind** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **size** () const noexcept
- constexpr [basic\\_string\\_view](#) **substr** (size\_type \_\_pos=0, size\_type \_\_n=npow) const noexcept(false)
- constexpr void **swap** ([basic\\_string\\_view](#) & \_\_sv) noexcept



## Static Public Attributes

- static constexpr size\_type **npos**

### 5.234.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::basic_string_view< _CharT, _Traits >
```

A non-owning reference to a string.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

A `basic_string_view` looks like this:

```
_CharT*    _M_str
size_t     _M_len
```

The documentation for this class was generated from the following files:

- [string\\_view](#)
- [bits/string\\_view.tcc](#)

## 5.235 std::experimental::filesystem::v1::basic\_string\_view< \_CharT, \_Traits > Class Template Reference

```
#include <>>
```

### Public Types

- using **const\_iterator**
- using **const\_pointer**
- using **const\_reference**
- using **const\_reverse\_iterator**
- using **difference\_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse\_iterator**
- using **size\_type**
- using **traits\_type**
- using **value\_type**

### Public Member Functions

- constexpr **basic\_string\_view** (const \_CharT \*\_\_str) noexcept
- constexpr **basic\_string\_view** (const \_CharT \*\_\_str, size\_type \_\_len) noexcept
- constexpr **basic\_string\_view** (const [basic\\_string\\_view](#) &) noexcept=default
- constexpr const\_reference **at** (size\_type \_\_pos) const
- constexpr const\_reference **back** () const noexcept
- constexpr const\_iterator **begin** () const noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept

- constexpr int **compare** ([basic\\_string\\_view](#) \_\_str) const noexcept
- constexpr int **compare** (const \_CharT \*\_\_str) const noexcept
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, [basic\\_string\\_view](#) \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, [basic\\_string\\_view](#) \_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_str, size\_type \_\_n2) const noexcept(false)
- constexpr size\_type **copy** (\_CharT \*\_\_str, size\_type \_\_n, size\_type \_\_pos=0) const
- constexpr [const\\_reverse\\_iterator](#) **cbegin** () const noexcept
- constexpr [const\\_reverse\\_iterator](#) **crend** () const noexcept
- constexpr const\_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr const\_reference **front** () const noexcept
- constexpr size\_type **length** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- constexpr [basic\\_string\\_view](#) & **operator=** (const [basic\\_string\\_view](#) &) noexcept=default
- constexpr const\_reference **operator[]** (size\_type \_\_pos) const noexcept
- constexpr [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- constexpr void **remove\_prefix** (size\_type \_\_n) noexcept
- constexpr void **remove\_suffix** (size\_type \_\_n) noexcept
- constexpr [const\\_reverse\\_iterator](#) **rend** () const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **rfind** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **size** () const noexcept
- constexpr [basic\\_string\\_view](#) **substr** (size\_type \_\_pos=0, size\_type \_\_n=npow) const noexcept(false)
- constexpr void **swap** ([basic\\_string\\_view](#) & \_\_sv) noexcept

## Static Public Attributes

- static constexpr size\_type **npos**

### 5.235.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits >
```

A non-owning reference to a string.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

A `basic_string_view` looks like this:

```
_CharT*    _M_str
size_t     _M_len
```

The documentation for this class was generated from the following files:

- [string\\_view](#)
- [bits/string\\_view.tcc](#)

## 5.236 std::experimental::fundamentals\_v1::basic\_string\_view< \_CharT, \_Traits > Class Template Reference

```
#include <>>
```

### Public Types

- using **const\_iterator**
- using **const\_pointer**
- using **const\_reference**
- using **const\_reverse\_iterator**
- using **difference\_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse\_iterator**
- using **size\_type**
- using **traits\_type**
- using **value\_type**

### Public Member Functions

- constexpr **basic\_string\_view** (const \_CharT \*\_\_str)
- constexpr **basic\_string\_view** (const \_CharT \*\_\_str, size\_type \_\_len)
- template<typename \_Allocator>  
  **basic\_string\_view** (const [basic\\_string](#)< \_CharT, \_Traits, \_Allocator > &\_\_str) noexcept
- constexpr **basic\_string\_view** (const [basic\\_string\\_view](#) &) noexcept=default
- constexpr const \_CharT & **at** (size\_type \_\_pos) const
- constexpr const \_CharT & **back** () const
- constexpr const\_iterator **begin** () const noexcept

- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr int **compare** (basic\_string\_view \_\_str) const noexcept
- constexpr int **compare** (const \_CharT \* \_\_str) const noexcept
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, basic\_string\_view \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, basic\_string\_view \_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \* \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \* \_\_str, size\_type \_\_n2) const
- size\_type **copy** (\_CharT \* \_\_str, size\_type \_\_n, size\_type \_\_pos=0) const
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- constexpr const \_CharT \* **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr const \_CharT & **front** () const
- constexpr size\_type **length** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- template<typename \_Allocator>  
operator basic\_string<\_CharT, \_Traits, \_Allocator> () const
- basic\_string\_view & operator= (const basic\_string\_view &) noexcept=default
- constexpr const \_CharT & operator[] (size\_type \_\_pos) const
- const\_reverse\_iterator **rbegin** () const noexcept
- constexpr void **remove\_prefix** (size\_type \_\_n)
- constexpr void **remove\_suffix** (size\_type \_\_n)
- const\_reverse\_iterator **rend** () const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **rfind** (basic\_string\_view \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **size** () const noexcept

- constexpr [basic\\_string\\_view](#) **substr** (size\_type \_\_pos=0, size\_type \_\_n=npos) const
- constexpr void **swap** ([basic\\_string\\_view](#) &\_\_sv) noexcept
- template<typename \_Allocator = std::allocator<\_CharT>>  
[basic\\_string](#)<\_CharT, \_Traits, \_Allocator> **to\_string** (const \_Allocator &\_\_alloc=\_Allocator()) const

### Static Public Attributes

- static constexpr size\_type **npos**

#### 5.236.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >
```

A non-owning reference to a string.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

A `basic_string_view` looks like this:

```
_CharT*    _M_str
size_t     _M_len
```

The documentation for this class was generated from the following files:

- [experimental/string\\_view](#)
- [experimental/bits/string\\_view.tcc](#)

### 5.237 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

```
#include <sstream>
```

Inheritance diagram for `std::basic_stringbuf< _CharT, _Traits, _Alloc >`:



### Public Types

- typedef \_\_string\_type::size\_type **\_\_size\_type**

- typedef [basic\\_streambuf](#)< char\_type, traits\_type > **\_\_streambuf\_type**
- typedef [basic\\_string](#)< char\_type, \_Traits, \_Alloc > **\_\_string\_type**
- typedef \_Alloc **allocator\_type**
- typedef \_CharT **char\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef \_Traits **traits\_type**

## Public Member Functions

- [basic\\_stringbuf](#) ()
- [basic\\_stringbuf](#) ([\\_\\_string\\_type](#) &&\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [basic\\_stringbuf](#) ([basic\\_stringbuf](#) &&\_\_rhs)
- [basic\\_stringbuf](#) ([basic\\_stringbuf](#) &&\_\_rhs, const allocator\_type &\_\_a)
- [basic\\_stringbuf](#) (const [\\_\\_string\\_type](#) &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [basic\\_stringbuf](#) (const allocator\_type &\_\_a)
- template<typename \_SAlloc>  
[basic\\_stringbuf](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > &\_\_s, const allocator\_type &\_\_a)
- template<typename \_SAlloc>  
[basic\\_stringbuf](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > &\_\_s, [ios\\_base::openmode](#) \_\_mode, const allocator\_type &\_\_a)
- template<typename \_SAlloc>  
[basic\\_stringbuf](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [basic\\_stringbuf](#) (const [basic\\_stringbuf](#) &)=delete
- [basic\\_stringbuf](#) ([ios\\_base::openmode](#) \_\_mode)
- [basic\\_stringbuf](#) ([ios\\_base::openmode](#) \_\_mode, const allocator\_type &\_\_a)
- allocator\_type [get\\_allocator](#) () const noexcept
- [locale](#) [getloc](#) () const
- [streamsize](#) [in\\_avail](#) ()
- [basic\\_stringbuf](#) & [operator=](#) ([basic\\_stringbuf](#) &&\_\_rhs)
- [basic\\_stringbuf](#) & [operator=](#) (const [basic\\_stringbuf](#) &)=delete
- [locale](#) [pubimbue](#) (const [locale](#) &\_\_loc)
- int\_type [sbumpc](#) ()
- int\_type [sgetc](#) ()
- [streamsize](#) [sgetn](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- int\_type [snextc](#) ()
- int\_type [sputbackc](#) (char\_type \_\_c)
- int\_type [sputc](#) (char\_type \_\_c)
- [streamsize](#) [sputn](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_string\\_type](#) [str](#) () &&
- [\\_\\_string\\_type](#) [str](#) () const &
- void [str](#) ([\\_\\_string\\_type](#) &&\_\_s)
- void [str](#) (const [\\_\\_string\\_type](#) &\_\_s)
- template<\_\_allocator\_like \_SAlloc>  
[basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > [str](#) (const \_SAlloc &\_\_sa) const
- template<\_\_allocator\_like \_SAlloc>  
requires (!is\_same\_v<\_SAlloc, \_Alloc>)  
void [str](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > &\_\_s)
- int\_type [sungetc](#) ()
- void [swap](#) ([basic\\_stringbuf](#) &\_\_rhs) noexcept(\_Noexcept\_swap::value)

- [basic\\_string\\_view](#)< char\_type, traits\_type > **view** () const noexcept
- [basic\\_streambuf](#) \* [pubsetbuf](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- pos\_type [pubseekoff](#) (off\_type \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- pos\_type [pubseekpos](#) (pos\_type \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- int [pubsync](#) ()

### Protected Member Functions

- void [\\_\\_safe\\_gbump](#) ([streamsize](#) \_\_n)
  - void [\\_\\_safe\\_pbump](#) ([streamsize](#) \_\_n)
  - void [\\_M\\_pbump](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend, off\_type \_\_off)
  - void [\\_M\\_stringbuf\\_init](#) ([ios\\_base::openmode](#) \_\_mode)
  - void [\\_M\\_sync](#) (char\_type \* \_\_base, \_\_size\_type \_\_i, \_\_size\_type \_\_o)
  - void [\\_M\\_update\\_egptr](#) ()
  - void [gbump](#) (int \_\_n)
  - virtual void [imbue](#) (const [locale](#) & \_\_loc)
  - virtual int\_type [overflow](#) (int\_type \_\_c=[traits\\_type::eof\(\)](#))
  - virtual int\_type [pbackfail](#) (int\_type \_\_c=[traits\\_type::eof\(\)](#))
  - void [pbump](#) (int \_\_n)
  - virtual pos\_type [seekoff](#) (off\_type \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - virtual pos\_type [seekpos](#) (pos\_type \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - virtual [\\_\\_streambuf\\_type](#) \* [setbuf](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
  - void [setg](#) (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
  - void [setp](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
  - virtual [streamsize](#) [showmanyc](#) ()
  - void [swap](#) ([basic\\_streambuf](#) & \_\_sb)
  - virtual int [sync](#) ()
  - virtual int\_type [uflow](#) ()
  - virtual int\_type [underflow](#) ()
  - virtual [streamsize](#) [xsgetn](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
  - virtual [streamsize](#) [xspn](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)
- 
- char\_type \* [eback](#) () const
  - char\_type \* [gptr](#) () const
  - char\_type \* [egptr](#) () const
- 
- char\_type \* [pbase](#) () const
  - char\_type \* [pptr](#) () const
  - char\_type \* [epptr](#) () const

### Protected Attributes

- [locale](#) [\\_M\\_buf\\_locale](#)
- char\_type \* [\\_M\\_in\\_beg](#)
- char\_type \* [\\_M\\_in\\_cur](#)
- char\_type \* [\\_M\\_in\\_end](#)
- [ios\\_base::openmode](#) [\\_M\\_mode](#)
- char\_type \* [\\_M\\_out\\_beg](#)
- char\_type \* [\\_M\\_out\\_cur](#)
- char\_type \* [\\_M\\_out\\_end](#)
- [\\_\\_string\\_type](#) [\\_M\\_string](#)

### 5.237.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringbuf< _CharT, _Traits, _Alloc >
```

The actual work of input and output (for std::string).

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.)

For this class, open modes (of type `ios_base::openmode`) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

### 5.237.2 Constructor & Destructor Documentation

#### basic\_stringbuf() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf () [inline]
```

Starts with an empty string buffer.

The default constructor initializes the parent class using its own default ctor.

#### basic\_stringbuf() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
    ios_base::openmode __mode) [inline], [explicit]
```

Starts with an empty string buffer.

#### Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

The default constructor initializes the parent class using its own default ctor.

#### basic\_stringbuf() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
    const __string_type & __str,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Starts with an existing string buffer.

#### Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

This constructor initializes the parent class using its own default ctor.



### 5.237.3 Member Function Documentation

#### eback()

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic\_streambuf< \_CharT, \_Traits >::eback \(\) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::pbackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

#### egptr()

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic\_streambuf< \_CharT, \_Traits >::egptr \(\) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::underflow\(\)](#), [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >::underflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#), and [xsgetn\(\)](#).

#### epptr()

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic\_streambuf< \_CharT, \_Traits >::epptr \(\) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**gbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_stringbuf<_CharT, _Traits, _Alloc>::gbump (
    int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↔</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the read position without returning any data.

Referenced by `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

**gptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::xsgetn()`, and `xsgetn()`.

**imbue()**

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf<_CharT, _Traits>::imbue (
    const locale & __loc) [inline], [protected], [virtual], [inherited]
```

Changes translations.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, and `std::basic_filebuf< wchar_t >`.

**in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
Looking ahead into the stream.
```

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**overflow()**

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::overflow (
    int_type __c = traits_type::eof()) [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                 |                                     |
|-----------------|-------------------------------------|
| <code>_↵</code> | An additional character to consume. |
| <code>_c</code> |                                     |

**Returns**

`eof()` to indicate failure, something else (usually `_c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `_c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::epptr()`, `std::basic_streambuf< _CharT, _Traits >::in`, `std::max()`, `std::min()`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pbump()`, `std::basic_streambuf< _CharT, _Traits >::pptr()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_streambuf< _CharT, _Traits >::setg()`.

**pbackfail()**

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::pbackfail (
    int_type __c = traits_type::eof()) [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_↵</code> | The character to be inserted back into the sequence. |
| <code>_c</code> |                                                      |

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::gbump()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::ios_base::out`.

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::sync()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
    int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`.

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__off</code>  | Offset.                       |
| <code>__way</code>  | Value for ios_base::seekdir.  |
| <code>__mode</code> | Value for ios_base::openmode. |

Calls virtual seekoff function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

**pubsetbuf()**

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

**pubsync()**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]
```

Calls virtual `sync` function.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**sbumpc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`.

**seekoff()**

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::e`, `std::ios_base::end`, `std::basic_streambuf< _CharT, _Traits >::eptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::ios_base::in`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pptr()`, and `std::basic_streambuf< _CharT, _Traits >::setg()`.

**seekpos()**

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [\\_M\\_mode](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::eback\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::egptr\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::epptr\(\)](#), [std::ios\\_base::in](#), [std::ios\\_base::out](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pbase\(\)](#) and [std::basic\\_streambuf< \\_CharT, \\_Traits >::setg\(\)](#).

**setbuf()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual __streambuf_type * std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf (
    char_type * __s,
    streamsize __n) [inline], [protected], [virtual]
```

Manipulates the buffer.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <code>↵<br/>__s</code> | Pointer to a buffer area.  |
| <code>↵<br/>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no buffer has already been created, and both `__s` and `__n` are non-zero, then `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.↵buffering> for more.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |



|                     |            |
|---------------------|------------|
| <code>__gend</code> | A pointer. |
|---------------------|------------|

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::wbuffer\\_convert\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#).

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setp (
    char_type * __pbeg,
    char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::wbuffer\\_convert\(\)](#).

**sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#).

**sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

## Parameters

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

**showmanyc()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc () [inline], [protected], [virtual]
```

Investigating the data available.

## Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

## Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

**snextc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or `eof`.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< char_type, traits_type >::ignore()`, `std::basic_istream< char_type, traits_type >::operator>>()`, and `std::basic_istream< char_type, traits_type >::putback()`.

**sputbackc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
    char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputc (
    char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
    const char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**str()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str () const & [inline], [nodiscard]
```

Copying out the string buffer.

**Returns**

A copy of one of the underlying sequences.

If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1

**str()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (
    const __string_type & __s) [inline]
```

Setting a new buffer.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Deallocates any previous stored sequence, then copies *s* to use as a new one.

**sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream< char_type, traits_type >::sentry::sentry()`.

**sync()**

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync () [inline], [protected], [virtual],
[inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_filebuf< wchar_t >`, and `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`.

**uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Referenced by `xsgetn()`.

**underflow()**

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::underflow () [protected], [virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::ios_base::in`.

**xsgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
    char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< \\_CharT, std::char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), [std::basic\\_filebuf< char >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), and [std::basic\\_filebuf< wchar\\_t >](#).  
References [egptr\(\)](#), [gptr\(\)](#), [std::min\(\)](#), and [uflow\(\)](#).

**xspn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xspn (
    const char_type * __s,
    streamsize __n)    [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< \\_CharT, std::char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), [std::basic\\_filebuf< char >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), and [std::basic\\_filebuf< wchar\\_t >](#).  
References [eptr\(\)](#), [std::min\(\)](#), [overflow\(\)](#), and [pptr\(\)](#).

**5.237.4 Member Data Documentation****M\_buf\_locale**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale    [protected], [inherited]
```

Current locale setting.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::basic\\_filebuf\(\)](#).

**M\_in\_beg**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg    [protected], [inherited]
```

Start of get area.

**\_M\_in\_cur**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
Current read area.
```

**\_M\_in\_end**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
End of get area.
```

**\_M\_mode**

```
template<typename _CharT, typename _Traits, typename _Alloc>
ios\_base::openmode std::basic\_stringbuf< _CharT, _Traits, _Alloc >::_M_mode [protected]
Place to stash in || out || in | out settings for current stringbuf.
Referenced by overflow\(\), pbackfail\(\), seekoff\(\), seekpos\(\), and underflow\(\).
```

**\_M\_out\_beg**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
Start of put area.
```

**\_M\_out\_cur**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
Current put area.
```

**\_M\_out\_end**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
End of put area.
```

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)
- [sstream.tcc](#)

**5.238 [std::basic\\_stringstream](#)< \_CharT, \_Traits, \_Alloc > Class Template Reference**

```
#include <sstream>
```

Inheritance diagram for std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`



- enum [event](#) { [erase\\_event](#) , [imbue\\_event](#) , [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
- typedef traits\_type::int\_type [int\\_type](#)
- typedef \_ios\_ostate [ostate](#)
- typedef traits\_type::off\_type [off\\_type](#)
- typedef \_ios\_Openmode [openmode](#)
- typedef traits\_type::pos\_type [pos\\_type](#)
- typedef \_ios\_Seekdir [seekdir](#)
- typedef \_Traits [traits\\_type](#)

## Public Member Functions

- [basic\\_stringstream](#) ()
- [basic\\_stringstream](#) (\_\_string\_type && \_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in|ios\\_base::out](#))
- [basic\\_stringstream](#) ([basic\\_stringstream](#) && \_\_rhs)
- [basic\\_stringstream](#) (const \_\_string\_type & \_\_str, [ios\\_base::openmode](#) \_\_m=[ios\\_base::out|ios\\_base::in](#))
- template<typename \_SAlloc>  
[basic\\_stringstream](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > & \_\_str, const allocator\_type & \_\_a)
- template<typename \_SAlloc>  
[basic\\_stringstream](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > & \_\_str, [ios\\_base::openmode](#) \_\_mode, const allocator\_type & \_\_a)
- template<typename \_SAlloc>  
[basic\\_stringstream](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > & \_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in|ios\\_base::out](#))
- [basic\\_stringstream](#) (const [basic\\_stringstream](#) &)=delete
- [basic\\_stringstream](#) ([ios\\_base::openmode](#) \_\_m)
- [basic\\_stringstream](#) ([ios\\_base::openmode](#) \_\_mode, const allocator\_type & \_\_a)
- ~[basic\\_stringstream](#) ()
- template<typename \_ValueT>  
[basic\\_istream](#)< \_CharT, \_Traits > & [\\_M\\_extract](#) (\_ValueT & \_\_v)
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename \_ValueT>  
[basic\\_ostream](#)< \_CharT, \_Traits > & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- \_\_ostream\_type & [flush](#) ()
- [streamsize](#) [gcount](#) () const
- [basic\\_istream](#)< char > & [getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [basic\\_istream](#)< wchar\_t > & [getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const

- [basic\\_istream](#)< char > & **ignore** (streamsize \_\_n)
  - [basic\\_istream](#)< wchar\_t > & **ignore** (streamsize \_\_n)
  - [basic\\_istream](#)< char > & **ignore** (streamsize \_\_n, int\_type \_\_delim)
  - [basic\\_istream](#)< wchar\_t > & **ignore** (streamsize \_\_n, int\_type \_\_delim)
  - [locale imbue](#) (const [locale](#) &\_\_loc)
  - long & [iword](#) (int \_\_ix)
  - char [narrow](#) (char\_type \_\_c, char \_\_default) const
  - [\\_\\_ostream\\_type](#) & **operator<<** ([\\_\\_streambuf\\_type](#) \*\_\_sb)
  - [\\_\\_ostream\\_type](#) & **operator<<** (const void \*\_\_p)
  - [\\_\\_ostream\\_type](#) & **operator<<** (nullptr\_t)
  - [basic\\_stringstream](#) & **operator=** ([basic\\_stringstream](#) &&\_\_rhs)
  - [basic\\_stringstream](#) & **operator=** (const [basic\\_stringstream](#) &)=delete
  - [\\_\\_istream\\_type](#) & **operator>>** ([\\_\\_streambuf\\_type](#) \*\_\_sb)
  - [\\_\\_istream\\_type](#) & **operator>>** (void \*&\_\_p)
  - [streamsize precision](#) () const
  - [streamsize precision](#) (streamsize \_\_prec)
  - void \*& [pword](#) (int \_\_ix)
  - [basic\\_streambuf](#)<\_CharT, \_Traits> \* [rdbuf](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
  - [\\_\\_stringbuf\\_type](#) \* [rdbuf](#) () const
  - [iostate rdstate](#) () const
  - void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
  - [\\_\\_ostream\\_type](#) & [seekp](#) (off\_type, [ios\\_base::seekdir](#))
  - [\\_\\_ostream\\_type](#) & [seekp](#) (pos\_type)
  - [fmtflags setf](#) (fmtflags \_\_fmtfl)
  - [fmtflags setf](#) (fmtflags \_\_fmtfl, [fmtflags](#) \_\_mask)
  - void [setstate](#) ([iostate](#) \_\_state)
  - [\\_\\_string\\_type str](#) () &&
  - [\\_\\_string\\_type str](#) () const &
  - void **str** ([\\_\\_string\\_type](#) &&\_\_s)
  - void **str** (const [\\_\\_string\\_type](#) &\_\_s)
  - template<\_\_allocator\_like \_SAlloc>  
[basic\\_string](#)<\_CharT, \_Traits, \_SAlloc> **str** (const \_SAlloc &\_\_sa) const
  - template<\_\_allocator\_like \_SAlloc>  
requires (is\_same\_v<\_SAlloc, \_Alloc>)  
void **str** (const [basic\\_string](#)<\_CharT, \_Traits, \_SAlloc> &\_\_s)
  - void **swap** ([basic\\_stringstream](#) &\_\_rhs)
  - pos\_type [tellp](#) ()
  - [basic\\_ostream](#)<\_CharT, \_Traits> \* [tie](#) () const
  - [basic\\_ostream](#)<\_CharT, \_Traits> \* [tie](#) ([basic\\_ostream](#)<\_CharT, \_Traits> \*\_\_tiestr)
  - void [unsetf](#) (fmtflags \_\_mask)
  - [basic\\_string\\_view](#)< char\_type, traits\_type> **view** () const noexcept
  - char\_type [widen](#) (char \_\_c) const
  - [streamsize width](#) () const
  - [streamsize width](#) (streamsize \_\_wide)
- 
- [\\_\\_istream\\_type](#) & **operator>>** ([\\_\\_istream\\_type](#) &(\*\_\_pf)([\\_\\_istream\\_type](#) &))
  - [\\_\\_istream\\_type](#) & **operator>>** ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
  - [\\_\\_istream\\_type](#) & **operator>>** ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

## Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`

## Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`

- [operator bool](#) () const
- [bool operator!](#) () const
- [\\_\\_ostream\\_type & operator<<](#) ( [\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
- [\\_\\_ostream\\_type & operator<<](#) ( [\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
- [\\_\\_ostream\\_type & operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type & operator<<](#) (long \_\_n)
- [\\_\\_ostream\\_type & operator<<](#) (unsigned long \_\_n)
- [\\_\\_ostream\\_type & operator<<](#) (bool \_\_n)
- [\\_\\_ostream\\_type & operator<<](#) (short \_\_n)
- [\\_\\_ostream\\_type & operator<<](#) (unsigned short \_\_n)
- [\\_\\_ostream\\_type & operator<<](#) (int \_\_n)
- [\\_\\_ostream\\_type & operator<<](#) (unsigned int \_\_n)
- [\\_\\_ostream\\_type & operator<<](#) (long long \_\_n)
- [\\_\\_ostream\\_type & operator<<](#) (unsigned long long \_\_n)
- [\\_\\_ostream\\_type & operator<<](#) (double \_\_f)
- [\\_\\_ostream\\_type & operator<<](#) (float \_\_f)
- [\\_\\_ostream\\_type & operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type & put](#) (char\_type \_\_c)
- [\\_\\_ostream\\_type & write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Public Attributes

- class [\\_\\_attribute](#)(([\\_\\_abi\\_tag](#)("cxx11")) failure typedef [\\_ios\\_Fmtflags](#) [fmtflags](#)

### Static Public Attributes

- static const [openmode](#) `__noreplace`
- static const [fmtflags](#) `adjustfield`
- static const [openmode](#) `app`
- static const [openmode](#) `ate`
- static const [iostate](#) `badbit`
- static const [fmtflags](#) `basefield`
- static const [seekdir](#) `beg`
- static const [openmode](#) `binary`
- static const [fmtflags](#) `boolalpha`
- static const [seekdir](#) `cur`
- static const [fmtflags](#) `dec`
- static const [seekdir](#) `end`
- static const [iostate](#) `eofbit`
- static const [iostate](#) `failbit`
- static const [fmtflags](#) `fixed`
- static const [fmtflags](#) `floatfield`
- static const [iostate](#) `goodbit`
- static const [fmtflags](#) `hex`
- static const [openmode](#) `in`
- static const [fmtflags](#) `internal`
- static const [fmtflags](#) `left`
- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) &\_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- template<typename \_ValueT>  
  [\\_\\_istream\\_type](#) & [\\_M\\_extract](#) (\_ValueT &\_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- template<typename \_ValueT>  
  [\\_\\_ostream\\_type](#) & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) &\_\_rhs) noexcept

- void **init** ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_iostream](#) &\_\_rhs)
- void **swap** ([basic\\_istream](#) &\_\_rhs)
- void **swap** ([basic\\_ostream](#) &\_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const [\\_\\_ctype\\_type](#) \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- char\_type [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [streamsize](#) [\\_M\\_gcount](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* [\\_M\\_num\\_get](#)
- const [\\_\\_num\\_put\\_type](#) \* [\\_M\\_num\\_put](#)
- [streamsize](#) [\\_M\\_precision](#)
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* [\\_M\\_streambuf](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [basic\\_ostream](#)<\_CharT, \_Traits> \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

### 5.238.1 Detailed Description

template<typename \_CharT, typename \_Traits, typename \_Alloc>  
class std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>

Controlling input and output for std::string.

Template Parameters

|                         |                                                                              |
|-------------------------|------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                    |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits</a> <_CharT>. |
| <a href="#">_Alloc</a>  | Allocator type, defaults to <a href="#">allocator</a> <_CharT>.              |

This class supports reading from and writing to objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_iostream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as *sb*.

### 5.238.2 Member Typedef Documentation

#### event\_callback

typedef void(\* [std::ios\\_base::event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i) [inherited]

The type of an event callback function.

**Parameters**

|                        |                                                        |
|------------------------|--------------------------------------------------------|
| <code>_↔<br/>_e</code> | One of the members of the event enum.                  |
| <code>_↔<br/>_b</code> | Reference to the <code>ios_base</code> object.         |
| <code>_↔<br/>_i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

**istate**

```
typedef _Ios_Iostate std::ios_base::istate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `istate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

**openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

### 5.238.3 Member Enumeration Documentation

#### event

enum `std::ios_base::event` [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

### 5.238.4 Constructor & Destructor Documentation

#### basic\_stringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream () [inline]
```

Default constructor starts with an empty string buffer.

Initializes `sb` using the mode `in|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer. That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

#### basic\_stringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream (
    ios_base::openmode __m) [inline], [explicit]
```

Starts with an empty string buffer.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__m</code> | Whether the buffer can read, or write, or both. |
|------------------|-------------------------------------------------|

Initializes `sb` using the mode from `__m`, and passes `&sb` to the base class initializer. Does not allocate any buffer. That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

#### basic\_stringstream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream (
    const __string_type & __str,
    ios_base::openmode __m = ios_base::out | ios_base::in) [inline], [explicit]
```

Starts with an existing string buffer.

#### Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__str</code> | A string to copy as a starting buffer.          |
| <code>__m</code>   | Whether the buffer can read, or write, or both. |

Initializes `sb` using `__str` and `__m`, and passes `&sb` to the base class initializer. That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

#### ~basic\_stringstream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream<_CharT, _Traits, _Alloc>::~~basic_stringstream () [inline]
```

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.



### 5.238.5 Member Function Documentation

#### **`_M_getloc()`**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_date\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_weekday\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::time\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::time\\_put<\\_CharT, \\_OutIter>::put\(\)](#).

#### **`bad()`**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic\_ios<\_CharT, \_Traits>::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

##### Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

#### **`clear()`**

```
template<typename _CharT, typename _Traits>
```

```
void std::basic\_ios<\_CharT, \_Traits>::clear (  
    iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

##### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::exceptions\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::unget\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

#### **`copyfmt()`**

```
template<typename _CharT, typename _Traits>
```

```
basic\_ios<\_CharT, \_Traits> & std::basic\_ios<\_CharT, \_Traits>::copyfmt (  
    const basic\_ios<\_CharT, \_Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

##### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof() const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::exceptions() const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::exceptions (
    iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
```

```
f.setstate (std::ios_base::badbit);

std::cerr << "Setting exception mask\n";
f.exceptions (std::ios_base::badbit);
}
```

### fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
Fast error checking.
```

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#), and [std::regex\\_traits< \\_Ch\\_type >::value\(\)](#).

### fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
Retrieves the empty character.
```

#### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by [copyfmt\(\)](#), [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#), and [std::subtract\\_with\\_carry\\_engine< uint\\_fast32\\_t, 24, 10, 24 >::operator\(\)\(\)](#).

### fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

### flags() [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::subtract\\_with\\_carry\\_engine<uint\\_fast32\\_t>::do\\_put\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator<<\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::subtract\\_with\\_carry\\_engine<uint\\_fast32\\_t>::do\\_put\(\)](#).

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

References [fmtflags](#).

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::pubsync\(\)](#).

**gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References [\\_M\\_gcount](#), and [std::ios\\_base::goodbit](#).

**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c) [inherited]
```

Simple extraction.

## Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

## Returns

`*this`

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

## Note

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::get (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

## Parameters

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

## Returns

`*this`

Returns `get(__s, __n, widen('\n'))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Simple multiple-character extraction.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>](#):

**getline()** [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                        |                                               |
|------------------------|-----------------------------------------------|
| <code>↵<br/>__s</code> | A character array in which to store the data. |
| <code>↵<br/>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s,__n,widen("\n"))`.

**getline()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References `_M_gcount`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**getline()** [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale::global C++ locale`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outiter>::do_put()`, `std::basic_ios<_CharT, _Traits>::do_get()`, `std::chrono::operator<<()`, `std::operator>>()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::__detail::operator>>()`.

**ignore()** [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
    void) [inherited]
```

Simple extraction.



**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#).

**ignore() [2/3]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

**ignore() [3/3]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim) [inherited]
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References `std::ios_base::getloc()`, `std::ios_base::imbue()`, and `rdbuf()`.

Referenced by `std::chrono::operator<<()`.

**init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
    basic_streambuf<_CharT, _Traits> * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_ios<char_type, traits_type>::basic_ios()`, and `std::basic_ostream<char_type, traits_type>::basic_ostream()`.

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References `iword()`.

Referenced by `iword()`.

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __C,
    char __dfault) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator"!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(* __pf )(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(* __pf )(__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

**operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↔        | A variable of builtin floating point type. |
| ↔        |                                            |
| ↔        |                                            |
| ↔        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↔        | A variable of builtin floating point type. |
| ↔        |                                            |
| ↔        |                                            |
| ↔        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    int __n) [inherited]
```

Integer arithmetic inserters.

## Parameters

|          |                                      |
|----------|--------------------------------------|
| ↔        | A variable of builtin integral type. |
| <i>n</i> |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(* __pf ) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<()** [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    short __n) [inherited]
```

Integer arithmetic inserters.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator<\\_CharT, \\_Traits>::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put<\\_CharT, \\_Outiter>::put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<()** [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.



**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type & (* __pf ) ( __istream_type & ) ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|                 |                                            |
|-----------------|--------------------------------------------|
| <code>↔</code>  | A variable of builtin floating point type. |
| <code>_↔</code> |                                            |
| <code>↔</code>  |                                            |
| <code>_↔</code> |                                            |
| <code>f</code>  |                                            |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|                 |                                            |
|-----------------|--------------------------------------------|
| <code>↔</code>  | A variable of builtin floating point type. |
| <code>_↔</code> |                                            |
| <code>↔</code>  |                                            |
| <code>_↔</code> |                                            |
| <code>f</code>  |                                            |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
    int & __n) [inherited]
```

Integer arithmetic extractors.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>()** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    ios_base &(* __pf ) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    short & __n) [inherited]
```

Integer arithmetic extractors.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::use_facet()`.

**operator>>()** [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↔</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\_ \leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [15/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\_ \leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [16/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\_ \leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [17/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

## Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_↔</code> | A variable of pointer type. |
| <code>_p</code> |                             |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

## Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline], [inherited]
```

Changing flags.

## Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

## Returns

The previous value of `precision()`.



**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c) [inherited]
```

Simple insertion.



## Parameters

|                                                                                   |                          |
|-----------------------------------------------------------------------------------|--------------------------|
|  | The character to insert. |
|  |                          |

## Returns

\*this

Tries to insert \_\_\_\_c.

### Note

This function is not overloaded on signed char and unsigned char.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits`

## putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c) [inherited]
```

### Unextracting a single character.

## Parameters

|                 |                                                   |
|-----------------|---------------------------------------------------|
| <code>_↵</code> | The character to push back into the input stream. |
| <code>_c</code> |                                                   |

## Returns

```
*this
```

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

### Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline], [inherited]
```

Access to void pointer array.

## Parameters

|                            |                       |
|----------------------------|-----------------------|
| $\_ \leftarrow$<br>$\_ ix$ | Index into the array. |
|----------------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_stringstream< _CharT, _Traits, _Alloc >::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

**rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios< char_type, traits_type >::bad()`, `clear()`, `std::basic_ios< char_type, traits_type >::eof()`,

`std::basic_ios< char_type, traits_type >::fail()`, `std::basic_ios< char_type, traits_type >::good()`, `std::basic_istream< _CharT, _Traits >::`

`std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type >::`

and `std::basic_istream< _CharT, _Traits >::unget()`.

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
    char_type * __s,
    streamsize __n)    [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
    char_type * __s,
    streamsize __n)    [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called A here:

- if `A == -1`, sets `eofbit` and extracts no characters
- if `A == 0`, extracts no characters

- if  $A > 0$ , extracts  $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::` and `std::basic_ios< _CharT, _Traits >::setstate()`.

### `register_callback()`

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

#### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

References `fmtflags`.

### `seekg()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    off_type __off,
    ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekoff(__off, __dir)`. If that function fails, sets `failbit`.

#### Note

This function first clears `eofbit`. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()` and `std::basic_ios< _CharT, _Traits >::setstate()`.

### `seekg()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekp() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
    off_type __off,
    ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekp() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
    pos_type __pos) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), and [std::ws\(\)](#).

**str()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringstream< _CharT, _Traits, _Alloc >::str () const & [inline], [nodiscard]
Copying out the string buffer.
```

**Returns**

`rdbuf ()->str ()`

**str()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (
    const __string_type & __s) [inline]
Setting a new buffer.
```

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Calls `rdbuf ()->str (s)`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
    void ) [inherited]
Synchronizing the stream buffer.
```

**Returns**

0 on success, -1 on failure

If `rdbuf ()` is a null pointer, returns -1.

Otherwise, calls `rdbuf ()->pubsync ()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount ()`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pubsync\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static], [inherited]
Interaction with the standard C I/O objects.
```

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg (
    void ) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp () [inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

References `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `copyfmt()`.

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie (
    basic_ostream<_CharT, _Traits> * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.



**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**ungetc()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ungetc (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References [fmtflags](#).

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::basic_ios<char_type, traits_type>::fill()`, `std::getline()`, `std::getline()`, `std::subtract_with_carry_engine<uint_fast32_t, std::tr2::operator>>()`, and `std::subtract_with_carry_engine<uint_fast32_t, 24, 10, 24>::seed()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outiter>::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::write (
    const char_type * __s,
    streamsize __n) [inherited]
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

\*this

Characters are copied from `___s` and inserted into the stream until one of the following happens:

- `___n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( )    [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.238.6 Member Data Documentation****`_M_gcount`**

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount    [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by [get\(\)](#), [get\(\)](#), [get\(\)](#), [get\(\)](#), [getline\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [putback\(\)](#), [read\(\)](#), [readsome\(\)](#), and [unget\(\)](#).

**`adjustfield`**

```
const fmtflags std::ios_base::adjustfield    [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Referenced by [std::num\\_put< \\_CharT, \\_Outlttr >::do\\_put\(\)](#), [std::internal\(\)](#), [std::left\(\)](#), and [std::right\(\)](#).

**`app`**

```
const openmode std::ios_base::app    [static], [inherited]
```

Seek to end before each write.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#).

**`ate`**

```
const openmode std::ios_base::ate    [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#).

**badbit**

```
const ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<char_type, traits_type>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<char_type, traits_type>::seekg()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_istream<char_type, traits_type>::tellg()`, and `std::ws()`.

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::oct()`.

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf<_CharT, _Traits>::seekpos()`.

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits>::seekoff()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`.

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::open\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_date\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_time\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::read\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::read\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::seekg\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_weekday\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::read\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), and [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#).

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), and [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#).

`std::basic_ostream< char_type, traits_type >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::ws()` and `std::ws()`.

## hex

const `fmtflags` `std::ios_base::hex` [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::hex()`.

## in

const `openmode` `std::ios_base::in` [static], [inherited]

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::xsgetn()` and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

## internal

const `fmtflags` `std::ios_base::internal` [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by `std::internal()`.

## left

const `fmtflags` `std::ios_base::left` [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

## oct

const `fmtflags` `std::ios_base::oct` [static], [inherited]

Converts integer input or generates integer output in octal base.

Referenced by `std::oct()`.

## out

const `openmode` `std::ios_base::out` [static], [inherited]

Open for output. Default for `ofstream` and `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

## right

const `fmtflags` `std::ios_base::right` [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.) Referenced by [std::right\(\)](#).

### scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

### showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

### showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

### showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

### skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), and [std::skipws\(\)](#).

### trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

### unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

### uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)

## 5.239 std::bernoulli\_distribution Class Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef bool [result\\_type](#)

### Public Member Functions

- [bernoulli\\_distribution](#) ()
- [bernoulli\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [bernoulli\\_distribution](#) (double \_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double [p](#) () const
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

### Friends

- bool [operator==](#) (const [bernoulli\\_distribution](#) &\_\_d1, const [bernoulli\\_distribution](#) &\_\_d2)

#### 5.239.1 Detailed Description

A Bernoulli random number distribution.

Generates a sequence of true and false values with likelihood  $p$  that true will come up and  $(1 - p)$  that false will appear.

Since

C++11

#### 5.239.2 Member Typedef Documentation

##### result\_type

```
typedef bool std::bernoulli_distribution::result_type
```

The type of the range of the distribution.



### 5.239.3 Constructor & Destructor Documentation

#### **bernoulli\_distribution()** [1/2]

```
std::bernoulli_distribution::bernoulli_distribution () [inline]
```

Constructs a Bernoulli distribution with likelihood 0.5.

References [bernoulli\\_distribution\(\)](#).

Referenced by [bernoulli\\_distribution\(\)](#), and [operator==](#).

#### **bernoulli\_distribution()** [2/2]

```
std::bernoulli_distribution::bernoulli_distribution (
    double __p) [inline], [explicit]
```

Constructs a Bernoulli distribution with likelihood p.

#### Parameters

|                                              |                                                                                         |
|----------------------------------------------|-----------------------------------------------------------------------------------------|
| <a href="#"><math>\leftrightarrow</math></a> | [IN] The likelihood of a true result being returned. Must be in the interval $[0, 1]$ . |
| <a href="#">_p</a>                           |                                                                                         |

### 5.239.4 Member Function Documentation

#### **max()**

```
result_type std::bernoulli_distribution::max () const [inline]
```

Returns the least upper bound value of the distribution.

References [std::numeric\\_limits<\\_Tp>::max\(\)](#).

#### **min()**

```
result_type std::bernoulli_distribution::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

References [std::numeric\\_limits<\\_Tp>::min\(\)](#).

#### **operator>()**

```
template<typename _UniformRandomNumberGenerator>
result_type std::bernoulli_distribution::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

References [operator>\(\)](#).

Referenced by [operator>\(\)](#).

#### **p()**

```
double std::bernoulli_distribution::p () const [inline]
```

Returns the p parameter of the distribution.

#### **param()** [1/2]

```
param_type std::bernoulli_distribution::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

**param()** [2/2]

```
void std::bernoulli_distribution::param (
    const param\_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
void std::bernoulli_distribution::reset () [inline]
```

Resets the distribution state.

Does nothing for a Bernoulli distribution.

**5.239.5 Friends And Related Symbol Documentation****operator==**

```
bool operator== (
    const bernoulli\_distribution & __d1,
    const bernoulli\_distribution & __d2) [friend]
```

Return true if two Bernoulli distributions have the same parameters.

References [bernoulli\\_distribution\(\)](#).

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.240 `std::bidirectional_iterator_tag` Struct Reference**

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for `std::bidirectional_iterator_tag`:



#### 5.240.1 Detailed Description

Bidirectional iterators support a superset of forward iterator operations.  
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 5.241 `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference`

```
#include <point_iterators.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_↵`

Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >:



## Public Types

- typedef Const\_Pointer **const\_pointer**
- typedef Const\_Reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value\_Type **value\_type**

## Public Member Functions

- **bin\_search\_tree\_const\_it\_** (const [bin\\_search\\_tree\\_const\\_it\\_](#)< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other)
- **bin\_search\_tree\_const\_it\_** (const Node\_Pointer p\_nd=0)
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_](#)< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other) const

- `bool operator!= (const bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const`
- `const_reference operator* () const`
- `bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & operator++ ()`
- `bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > operator++ (int)`
- `bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & operator-- ()`
- `bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > operator-- (int)`
- `const_pointer operator-> () const`
- `bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & operator= (const bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)`
- `bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & operator= (const bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)`
- `bool operator== (const bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const`
- `bool operator== (const bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const`

#### Public Attributes

- `Node_Pointer m_p_nd`

#### Protected Member Functions

- `void dec (false_type)`
- `void dec (true_type)`
- `void inc (false_type)`
- `void inc (true_type)`

#### 5.241.1 Detailed Description

`template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>`  
`class \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_it\_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`

Const iterator.

The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)

#### 5.242 `\_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_ < Node, Const_Iterator, Iterator, _Alloc >` Class Template Reference

```
#include <node_iterators.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >`:



## Public Types

- typedef Const\_Iterator [const\\_reference](#)
- typedef [trivial\\_iterator\\_difference\\_type](#) [difference\\_type](#)
- typedef [trivial\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef [rebind\\_traits](#)< \_Alloc, [metadata\\_type](#) >::const\_reference [metadata\\_const\\_reference](#)
- typedef Node::metadata\_type [metadata\\_type](#)
- typedef Const\_Iterator [reference](#)
- typedef Const\_Iterator [value\\_type](#)

## Public Member Functions

- [bin\\_search\\_tree\\_const\\_node\\_it\\_](#) (const node\_pointer p\_nd=0)
- [bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) [get\\_l\\_child](#) () const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- [bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) [get\\_r\\_child](#) () const
- bool [operator!=](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) &other) const
- [const\\_reference](#) [operator\\*](#) () const
- bool [operator==](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) &other) const

## Public Attributes

- node\_pointer [m\\_p\\_nd](#)

### 5.242.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Const node iterator.

### 5.242.2 Member Typedef Documentation

#### [const\\_reference](#)

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::const_reference
```

Iterator's `__const` reference type.

**difference\_type**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial\_iterator\_difference\_type \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type
Difference type.
```

**iterator\_category**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial\_iterator\_tag \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_↵
Iterator, Iterator, _Alloc >::iterator_category
Category.
```

**metadata\_const\_reference**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef rebind\_traits<_Alloc,metadata\_type>::const_reference \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference
Const metadata reference type.
```

**metadata\_type**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Node::metadata_type \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_↵
Iterator, Iterator, _Alloc >::metadata_type
Metadata type.
```

**reference**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_Iterator,
Iterator, _Alloc >::reference
Iterator's reference type.
```

**value\_type**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_Iterator,
Iterator, _Alloc >::value_type
Iterator's value type.
```

**5.242.3 Member Function Documentation****get\_l\_child()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin\_search\_tree\_const\_node\_it\_< Node, Const_Iterator, Iterator, _Alloc > \_\_gnu\_pbds::detail::bin\_search\_tree\_con
Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]
Returns the \_\_const node iterator associated with the left node.
```

**get\_metadata()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
metadata\_const\_reference \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const [inline]
```

Metadata access.

### `get_r_child()`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]
Returns the __const node iterator associated with the right node.
```

### `operator!=(())`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
    const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
    other) const [inline]
Compares (negatively) to a different iterator object.
```

### `operator*()`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator,
_Alloc >::operator* () const [inline]
Access.
```

### `operator==(())`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator== (
    const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
    other) const [inline]
Compares to a different iterator object.
```

The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)

## 5.243 `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference`

```
#include <point_iterators.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer,`



Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >:



## Public Types

- typedef Const\_Pointer **const\_pointer**
- typedef Const\_Reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value\_Type **value\_type**

## Public Member Functions

- **bin\_search\_tree\_it\_** (const [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other)
- **bin\_search\_tree\_it\_** (const Node\_Pointer p\_nd=0)
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other) const
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >::reference **operator\*** () const
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔ Forward\_Iterator, \_Alloc > & **operator++** ()
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔ Forward\_Iterator, \_Alloc > **operator++** (int)
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔ Forward\_Iterator, \_Alloc > & **operator--** ()
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔ Forward\_Iterator, \_Alloc > **operator--** (int)
- [bin\\_search\\_tree\\_const\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >::pointer **operator->** () const
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔ Forward\_Iterator, \_Alloc > & **operator=** (const [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_↔ \_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other)
- [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↔ Forward\_Iterator, \_Alloc > & **operator=** (const [bin\\_search\\_tree\\_it\\_](#) < Node\_Pointer, Value\_Type, Pointer, Const\_↔ \_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)

- `bool operator==` (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, !Is\\_Forward\\_Iterator, \\_Alloc > &other](#)) const
- `bool operator==` (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc > &other](#)) const

#### Public Attributes

- Node\_Pointer `m_p_nd`

#### Protected Types

- typedef [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_←Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) `base_it_type`

#### Protected Member Functions

- void `dec` (false\_type)
- void `dec` (true\_type)
- void `inc` (false\_type)
- void `inc` (true\_type)

#### 5.243.1 Detailed Description

template<typename Node\_Pointer, typename Value\_Type, typename Pointer, typename Const\_Pointer, typename Reference, typename Const\_Reference, bool Is\_Forward\_Iterator, typename \_Alloc>  
 class `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`

Iterator.

The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)

#### 5.244 `__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class` Template Reference

```
#include <node_iterators.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >`:



## Public Types

- typedef Iterator [const\\_reference](#)
- typedef [trivial\\_iterator\\_difference\\_type](#) [difference\\_type](#)
- typedef [trivial\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef [rebind\\_traits](#)< [\\_Alloc](#), [metadata\\_type](#) >::[const\\_reference](#) [metadata\\_const\\_reference](#)
- typedef Node::metadata\_type [metadata\\_type](#)
- typedef Iterator [reference](#)
- typedef Iterator [value\\_type](#)

## Public Member Functions

- [bin\\_search\\_tree\\_node\\_it\\_](#) (const node\_pointer p\_nd=0)
- [bin\\_search\\_tree\\_node\\_it\\_](#)< Node, Const\_Iterator, Iterator, [\\_Alloc](#) > [get\\_l\\_child](#) () const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- [bin\\_search\\_tree\\_node\\_it\\_](#)< Node, Const\_Iterator, Iterator, [\\_Alloc](#) > [get\\_r\\_child](#) () const
- bool [operator!=](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_](#)< Node, Const\_Iterator, Iterator, [\\_Alloc](#) > &other) const
- Iterator [operator\\*](#) () const
- bool [operator==](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_](#)< Node, Const\_Iterator, Iterator, [\\_Alloc](#) > &other) const

## Public Attributes

- node\_pointer [m\\_p\\_nd](#)

### 5.244.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Node iterator.

### 5.244.2 Member Typedef Documentation

#### [const\\_reference](#)

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _↵
_Alloc >::const_reference
Iterator's __const reference type.
```

#### [difference\\_type](#)

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial\_iterator\_difference\_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type [inherited]
Difference type.
```

#### [iterator\\_category](#)

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial\_iterator\_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↵
Iterator, Iterator, _Alloc >::iterator_category [inherited]
Category.
```

**metadata\_const\_reference**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef rebind\_traits<_Alloc,metadata\_type>::const_reference \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference [inherited]
Const metadata reference type.
```

**metadata\_type**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Node::metadata_type \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_Iterator, Iterator, Iterator, _Alloc >::metadata_type [inherited]
Metadata type.
```

**reference**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const_Iterator, Iterator, _Alloc >::reference
Iterator's reference type.
```

**value\_type**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const_Iterator, Iterator, _Alloc >::value_type
Iterator's value type.
```

**5.244.3 Member Function Documentation****get\_l\_child()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin\_search\_tree\_node\_it\_< Node, Const_Iterator, Iterator, _Alloc > \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_<
Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]
Returns the node iterator associated with the left node.
```

**get\_metadata()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
metadata\_const\_reference \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const [inline], [inherited]
Metadata access.
```

**get\_r\_child()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin\_search\_tree\_node\_it\_< Node, Const_Iterator, Iterator, _Alloc > \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_<
Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]
Returns the node iterator associated with the right node.
```

**operator"!="()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
```

```
const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const [inline], [inherited]
```

Compares (negatively) to a different iterator object.

#### operator\*()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >↵
::operator* () const [inline]
```

Access.

#### operator==( )

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator==(
```

```
const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const [inline], [inherited]
```

Compares to a different iterator object.

The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)

## 5.245 \_\_gnu\_pbds::detail::bin\_search\_tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, Node, \_Alloc > Struct Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef [bin\\_search\\_tree\\_const\\_it\\_< typename node\\_alloc\\_traits::pointer, typename \[type\\\_traits::value\\\_type\]\(#\), typename \[type\\\_traits::pointer\]\(#\), typename \[type\\\_traits::const\\\_pointer\]\(#\), typename \[type\\\_traits::reference\]\(#\), typename \[type\\\_traits::const\\\_reference\]\(#\), false, \\_Alloc >](#) **const\_reverse\_iterator**
- typedef Node **node**
- typedef [bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, \[point\\\_const\\\_iterator\]\(#\), \[point\\\_iterator\]\(#\), \\_Alloc >](#) **node\_const\_iterator**
- typedef [bin\\_search\\_tree\\_node\\_it\\_< Node, \[point\\\_const\\\_iterator\]\(#\), \[point\\\_iterator\]\(#\), \\_Alloc >](#) **node\_iterator**
- typedef Node\_Update< [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update< \[node\\\_const\\\_iterator\]\(#\), \[node\\\_iterator\]\(#\), Cmp\\_Fn, \\_Alloc > \\* null\\_node\\_↵\\_update\\_pointer](#)
- typedef [bin\\_search\\_tree\\_const\\_it\\_< typename node\\_alloc\\_traits::pointer, typename \[type\\\_traits::value\\\_type\]\(#\), typename \[type\\\_traits::pointer\]\(#\), typename \[type\\\_traits::const\\\_pointer\]\(#\), typename \[type\\\_traits::reference\]\(#\), typename \[type\\\_traits::const\\\_reference\]\(#\), true, \\_Alloc >](#) **point\_const\_iterator**
- typedef [bin\\_search\\_tree\\_it\\_< typename node\\_alloc\\_traits::pointer, typename \[type\\\_traits::value\\\_type\]\(#\), type-  
name \[type\\\_traits::pointer\]\(#\), typename \[type\\\_traits::const\\\_pointer\]\(#\), typename \[type\\\_traits::reference\]\(#\), typename \[type\\\_traits::const\\\_reference\]\(#\), true, \\_Alloc >](#) **point\_iterator**
- typedef [bin\\_search\\_tree\\_it\\_< typename node\\_alloc\\_traits::pointer, typename \[type\\\_traits::value\\\_type\]\(#\), type-  
name \[type\\\_traits::pointer\]\(#\), typename \[type\\\_traits::const\\\_pointer\]\(#\), typename \[type\\\_traits::reference\]\(#\), typename \[type\\\_traits::const\\\_reference\]\(#\), false, \\_Alloc >](#) **reverse\_iterator**

### 5.245.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr,
class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
```

Binary search tree traits, primary template.

## 5.245.2 Member Typedef Documentation

`node_const_iterator`

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_
Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::
Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

5.246 `__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

## Public Types

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `Node` **node**
- typedef `Node` **node**
- typedef `bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `node_const_iterator` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_←_update_pointer`
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_←_update_pointer`
- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**
- typedef `const_reverse_iterator` **reverse\_iterator**

### 5.246.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class _Cmp_Fn, type-
name _Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >
```

Specialization.

### 5.246.2 Member Typedef Documentation

#### node\_const\_iterator [1/2]

```
typedef bin_search_tree_const_node_it< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::K
Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

#### node\_const\_iterator [2/2]

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class _Cmp_Fn,
typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin_search_tree_const_node_it< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::K
Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

## 5.247 \_\_gnu\_cxx::binary\_compose< \_Operation1, \_Operation2, \_Operation3 > Class Template Reference

```
#include <functional>
```

Inheritance diagram for \_\_gnu\_cxx::binary\_compose< \_Operation1, \_Operation2, \_Operation3 >:



### Public Types

- typedef \_Operation2::argument\_type [argument\\_type](#)

- `typedef _Operation1::result_type` [result\\_type](#)

### Public Member Functions

- **`binary_compose`** (`const _Operation1 &__x`, `const _Operation2 &__y`, `const _Operation3 &__z`)
- `_Operation1::result_type` **`operator()`** (`const typename _Operation2::argument_type &__x`) `const`

### Protected Attributes

- `_Operation1` **`_M_fn1`**
- `_Operation2` **`_M_fn2`**
- `_Operation3` **`_M_fn3`**

#### 5.247.1 Detailed Description

```
template<class _Operation1, class _Operation2, class _Operation3>
class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >
```

An [SGI extension](#) .

#### 5.247.2 Member Typedef Documentation

##### `argument_type`

```
typedef _Operation2::argument_type std::unary\_function< _Operation2::argument_type, _Operation1↵
::result_type >::argument_type [inherited]
argument_type is the type of the argument
```

##### `result_type`

```
typedef _Operation1::result_type std::unary\_function< _Operation2::argument_type, _Operation1↵
::result_type >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.248 `std::binary_function< _Arg1, _Arg2, _Result >` Struct Template Reference

```
#include <stl_function.h>
```



Inheritance diagram for `std::binary_function<_Arg1, _Arg2, _Result>`:



### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### 5.248.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
struct std::binary_function< _Arg1, _Arg2, _Result >
```

Helper for defining adaptable binary function objects.

**Deprecated** Deprecated in C++11, no longer in the standard since C++17.

### 5.248.2 Member Typedef Documentation

#### `first_argument_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type
first_argument_type is the type of the first argument
```

#### `result_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type
result_type is the return type
```

#### `second_argument_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.249 `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

```
#include <binary_heap_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `cond_dealtor< value_type, _Alloc >` **cond\_dealtor\_t**
- typedef `binary_heap_const_iterator< value_type, entry, simple_value, _Alloc >` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `__conditional_type< simple_value, value_type, pointer >::__type` **entry**
- typedef `rebind_traits< _Alloc, entry >::allocator_type` **entry\_allocator**
- typedef `entry_cmp< Value_Type, Cmp_Fn, _Alloc, is_simple< Value_Type >::value >::type` **entry\_cmp**
- typedef `rebind_traits< _Alloc, entry >::pointer` **entry\_pointer**
- typedef `const_iterator` **iterator**
- typedef `binary_heap_point_const_iterator< value_type, entry, simple_value, _Alloc >` **point\_const\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `__gnu_pbds::detail::resize_policy< typename _Alloc::size_type >` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **binary\_heap** (const `binary_heap` &)
- **binary\_heap** (const `cmp_fn` &)
- **iterator** **begin** ()
- **const\_iterator** **begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator** **end** ()
- **const\_iterator** **end** () const
- void **erase** (`point_iterator`)
- void **erase\_at** (`entry_pointer`, `size_type`, `false_type`)
- void **erase\_at** (`entry_pointer`, `size_type`, `true_type`)
- template<typename Pred>  
  `size_type` **erase\_if** (Pred)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- `size_type` **get\_new\_size\_for\_arbitrary** (`size_type`) const
- `size_type` **get\_new\_size\_for\_grow** () const
- `size_type` **get\_new\_size\_for\_shrink** () const
- bool **grow\_needed** (`size_type`) const
- void **join** (`binary_heap` &)
- `size_type` **max\_size** () const
- void **modify** (`point_iterator`, `const_reference`)
- void **notify\_arbitrary** (`size_type`)
- void **notify\_grow\_resize** ()
- void **notify\_shrink\_resize** ()
- void **pop** ()
- `point_iterator` **push** (`const_reference`)
- bool **resize\_needed\_for\_grow** (`size_type`) const

- bool **resize\_needed\_for\_shrink** (size\_type) const
- bool **shrink\_needed** (size\_type) const
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, [binary\\_heap](#) &)
- void **swap** ([binary\\_heap](#) &)
- void **swap** ([resize\\_policy](#)< \_Alloc::size\_type > &)
- const\_reference **top** () const

### Static Public Attributes

- static const \_Alloc::size\_type **min\_size**

### Protected Member Functions

- template<typename It>  
void **copy\_from\_range** (It, It)

#### 5.249.1 Detailed Description

**template<typename Value\_Type, typename Cmp\_Fn, typename \_Alloc>**  
**class** `__gnu_pbds::detail::binary_heap`< Value\_Type, Cmp\_Fn, \_Alloc >

Binary heaps composed of resize and compare policies.  
Based on CLRS.

The documentation for this class was generated from the following file:

- [binary\\_heap.hpp](#)

## 5.250 `__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class` Template Reference

```
#include <const_iterator.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >`:



### Public Types

- typedef [base\\_type::const\\_pointer](#) **const\_pointer**
- typedef [base\\_type::const\\_reference](#) **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**

- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

### Public Member Functions

- `binary_heap_const_iterator_()`
- `binary_heap_const_iterator_(const binary_heap_const_iterator_ &other)`
- `binary_heap_const_iterator_(entry_pointer p_e)`
- `bool operator!= (const binary_heap_const_iterator_ &other) const`
- `bool operator!= (const binary_heap_point_const_iterator_ &other) const`
- `const_reference operator* () const`
- `binary_heap_const_iterator_ & operator++ ()`
- `binary_heap_const_iterator_ operator++ (int)`
- `const_pointer operator-> () const`
- `bool operator== (const binary_heap_const_iterator_ &other) const`
- `bool operator== (const binary_heap_point_const_iterator_ &other) const`

### Public Attributes

- `entry_pointer m_p_e`

#### 5.250.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

#### 5.250.2 Member Typedef Documentation

##### `const_pointer`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::const_pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_pointer
Iterator's const pointer type.
```

##### `const_reference`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::const_reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_reference
Iterator's const reference type.
```

##### `difference_type`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef _Alloc::difference_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::difference_type
Difference type.
```

**iterator\_category**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef std::forward_iterator_tag __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
Category.
```

**pointer**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::pointer
Iterator's pointer type.
```

**reference**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::reference
Iterator's reference type.
```

**value\_type**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::value_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
Iterator's value type.
```

**5.250.3 Constructor & Destructor Documentation****binary\_heap\_const\_iterator\_()** [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_↵
heap_const_iterator_ () [inline]
Default constructor.
```

**binary\_heap\_const\_iterator\_()** [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_↵
heap_const_iterator_ (
    const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
[inline]
Copy constructor.
```

**5.250.4 Member Function Documentation****operator"!="()** [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=
(
    const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline]
Compares content (negatively) to a different iterator object.
```

**operator!=()** [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator!= (
    const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
```

const [inline], [inherited]

Compares content (negatively) to a different iterator object.

**operator\*()**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const [inline], [inherited]
```

Access.

**operator->()**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↵
_Alloc >::operator-> () const [inline], [inherited]
```

Access.

**operator==()** [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator==
(
    const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
```

const [inline]

Compares content to a different iterator object.

**operator==()** [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator== (
    const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
```

const [inline], [inherited]

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_/const\\_iterator.hpp](#)

## 5.251 \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc > Class Template Reference

```
#include <point_const_iterator.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc

>:



## Public Types

- typedef [rebind\\_traits](#)< [\\_Alloc](#), [value\\_type](#) >::const\_pointer [const\\_pointer](#)
- typedef [rebind\\_traits](#)< [\\_Alloc](#), [value\\_type](#) >::const\_reference [const\\_reference](#)
- typedef [trivial\\_iterator\\_difference\\_type](#) [difference\\_type](#)
- typedef [trivial\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef [rebind\\_traits](#)< [\\_Alloc](#), [value\\_type](#) >::pointer [pointer](#)
- typedef [rebind\\_traits](#)< [\\_Alloc](#), [value\\_type](#) >::reference [reference](#)
- typedef [Value\\_Type](#) [value\\_type](#)

## Public Member Functions

- [binary\\_heap\\_point\\_const\\_iterator\\_](#) ()
- [binary\\_heap\\_point\\_const\\_iterator\\_](#) (const [binary\\_heap\\_point\\_const\\_iterator\\_](#) &other)
- [binary\\_heap\\_point\\_const\\_iterator\\_](#) (entry\_pointer p\_e)
- bool [operator!=](#) (const [binary\\_heap\\_point\\_const\\_iterator\\_](#) &other) const
- [const\\_reference operator\\*](#) () const
- [const\\_pointer operator->](#) () const
- bool [operator==](#) (const [binary\\_heap\\_point\\_const\\_iterator\\_](#) &other) const

## Public Attributes

- entry\_pointer [m\\_p\\_e](#)

## Protected Types

- typedef [rebind\\_traits](#)< [\\_Alloc](#), [Entry](#) >::pointer [entry\\_pointer](#)



### 5.251.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

### 5.251.2 Member Typedef Documentation

#### const\_pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind\_traits<_Alloc, value\_type>::const_pointer \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_<
Value_Type, Entry, Simple, _Alloc >::const_pointer
Iterator's const pointer type.
```

#### const\_reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind\_traits<_Alloc, value\_type>::const_reference \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_<
Value_Type, Entry, Simple, _Alloc >::const_reference
Iterator's const reference type.
```

#### difference\_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef trivial\_iterator\_difference\_type \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_<
Value_Type, Entry, Simple, _Alloc >::difference_type
Difference type.
```

#### iterator\_category

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef trivial\_iterator\_tag \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
Category.
```

#### pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind\_traits<_Alloc, value\_type>::pointer \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_<
Value_Type, Entry, Simple, _Alloc >::pointer
Iterator's pointer type.
```

#### reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind\_traits<_Alloc, value\_type>::reference \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_<
Value_Type, Entry, Simple, _Alloc >::reference
Iterator's reference type.
```

#### value\_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef Value_Type \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value_Type, Entry,
Simple, _Alloc >::value_type
Iterator's value type.
```

### 5.251.3 Constructor & Destructor Documentation

#### `binary_heap_point_const_iterator_()` [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::binary_heap_point_const_iterator_ () [inline]
```

Default constructor.

#### `binary_heap_point_const_iterator_()` [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::binary_heap_point_const_iterator_ (
    const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
[inline]
```

Copy constructor.

### 5.251.4 Member Function Documentation

#### `operator"!="()`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator!= (
    const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline]
```

Compares content (negatively) to a different iterator object.

#### `operator*()`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const [inline]
```

Access.

#### `operator->()`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↵
_Alloc >::operator-> () const [inline]
```

Access.

#### `operator==()`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator== (
    const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_point\\_const\\_iterator.hpp](#)

### 5.252 \_\_gnu\_pbds::binary\_heap\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::binary\_heap\_tag:



#### 5.252.1 Detailed Description

Binary-heap (array-based).

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.253 std::binary\_negate<\_Predicate> Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::binary_negate<_Predicate>`:



#### Public Types

- `typedef _Predicate::first_argument_type` [first\\_argument\\_type](#)

- typedef bool [result\\_type](#)
- typedef `_Predicate::second_argument_type` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr **binary\_negate** (const `_Predicate` &\_\_x)
- constexpr bool **operator()** (const typename `_Predicate::first_argument_type` &\_\_x, const typename `_Predicate::second_argument_type` &\_\_y) const

### Protected Attributes

- `_Predicate_M_pred`

#### 5.253.1 Detailed Description

`template<typename _Predicate>`  
**class** `std::binary_negate<_Predicate>`

One of the [negation functors](#).

#### 5.253.2 Member Typedef Documentation

##### `first_argument_type`

typedef `_Predicate::first_argument_type` [std::binary\\_function](#)< `_Predicate::first_argument_type`, `_Predicate::second_argument_type`, bool >::`first_argument_type` [inherited]  
`first_argument_type` is the type of the first argument

##### `result_type`

typedef bool [std::binary\\_function](#)< `_Predicate::first_argument_type`, `_Predicate::second_argument_type`, bool >::`result_type` [inherited]  
`result_type` is the return type

##### `second_argument_type`

typedef `_Predicate::second_argument_type` [std::binary\\_function](#)< `_Predicate::first_argument_type`, `_Predicate::second_argument_type`, bool >::`second_argument_type` [inherited]  
`second_argument_type` is the type of the second argument

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.254 `std::binder1st<_Operation>` Class Template Reference

```
#include <binders.h>
```

Inheritance diagram for `std::binder1st<_Operation>`:



### Public Types

- typedef `_Operation::second_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

### Public Member Functions

- **binder1st** (const `_Operation` &\_\_x, const typename `_Operation::first_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::second_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::second_argument_type` &\_\_x) const

### Protected Attributes

- `_Operation` **op**
- `_Operation::first_argument_type` **value**

#### 5.254.1 Detailed Description

```
template<typename _Operation>
class std::binder1st<_Operation>
```

One of the [binder functors](#).

#### 5.254.2 Member Typedef Documentation

##### **argument\_type**

```
typedef _Operation::second_argument_type std::unary\_function<_Operation::second_argument_type, ↔
_Operation::result_type >::argument_type [inherited]
argument_type is the type of the argument
```

**result\_type**

```
typedef _Operation::result_type std::unary\_function< _Operation::second_argument_type, _Operation←
::result_type >::result_type [inherited]
```

result\_type is the return type

The documentation for this class was generated from the following file:

- [backward/binders.h](#)

**5.255 std::binder2nd< \_Operation > Class Template Reference**

```
#include <binders.h>
```

Inheritance diagram for std::binder2nd< \_Operation >:

**Public Types**

- typedef `_Operation::first_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

**Public Member Functions**

- **binder2nd** (const `_Operation` &\_\_x, const typename `_Operation::second_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::first_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::first_argument_type` &\_\_x) const

**Protected Attributes**

- `_Operation` **op**
- `_Operation::second_argument_type` **value**

**5.255.1 Detailed Description**

```
template<typename _Operation>
class std::binder2nd< _Operation >
```

One of the [binder functors](#).

### 5.255.2 Member Typedef Documentation

#### argument\_type

```
typedef _Operation::first_argument_type std::unary\_function< _Operation::first_argument_type, _↵  
Operation::result_type >::argument_type [inherited]  
argument_type is the type of the argument
```

#### result\_type

```
typedef _Operation::result_type std::unary\_function< _Operation::first_argument_type, _Operation↵  
::result_type >::result_type [inherited]  
result_type is the return type
```

The documentation for this class was generated from the following file:

- [backward/binders.h](#)

### 5.256 [std::binomial\\_distribution](#)< \_IntType > Class Template Reference

```
#include <random>
```

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef \_IntType [result\\_type](#)

#### Public Member Functions

- [binomial\\_distribution](#) (\_IntType \_\_t, double \_\_p=0.5)
- [binomial\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator](#)() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator](#)() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double [p](#) () const
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()
- \_IntType [t](#) () const

## Friends

- `template<typename _IntType1, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::binomial_distribution< _IntType1 > &__x)`
- `bool operator== (const binomial_distribution &__d1, const binomial_distribution &__d2)`
- `template<typename _IntType1, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::binomial_distribution< _IntType1 > &__x)`

### 5.256.1 Detailed Description

`template<typename _IntType = int>`  
**class** `std::binomial_distribution< _IntType >`

A discrete binomial random number distribution.

The formula for the binomial probability density function is  $p(i|t, p) = \binom{t}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Since

C++11

### 5.256.2 Member Typedef Documentation

#### result\_type

`template<typename _IntType = int>`  
`typedef _IntType std::binomial_distribution< _IntType >::result_type`  
 The type of the range of the distribution.

### 5.256.3 Member Function Documentation

#### max()

`template<typename _IntType = int>`  
`result_type std::binomial_distribution< _IntType >::max () const [inline]`  
 Returns the least upper bound value of the distribution.

#### min()

`template<typename _IntType = int>`  
`result_type std::binomial_distribution< _IntType >::min () const [inline]`  
 Returns the greatest lower bound value of the distribution.

#### operator>() [1/2]

`template<typename _IntType = int>`  
`template<typename _UniformRandomNumberGenerator>`  
`result_type std::binomial_distribution< _IntType >::operator() (`  
`_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Referenced by `operator>()`.

#### operator>() [2/2]

`template<typename _IntType>`  
`template<typename _UniformRandomNumberGenerator>`



```

binomial_distribution< _IntType >::result_type std::binomial_distribution< _IntType >::operator()
(
    _UniformRandomNumberGenerator & __urng,
    const param_type & __param)

```

A rejection algorithm when  $t * p \geq 8$  and a simple waiting time method - the second in the referenced book - otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_FUNCS` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

References `std::numeric_limits< _Tp >::epsilon()`, and `std::numeric_limits< _Tp >::max()`.

## p()

```

template<typename _IntType = int>
double std::binomial_distribution< _IntType >::p () const [inline]

```

Returns the distribution p parameter.

## param() [1/2]

```

template<typename _IntType = int>
param_type std::binomial_distribution< _IntType >::param () const [inline]

```

Returns the parameter set of the distribution.

## param() [2/2]

```

template<typename _IntType = int>
void std::binomial_distribution< _IntType >::param (
    const param_type & __param) [inline]

```

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

## reset()

```

template<typename _IntType = int>
void std::binomial_distribution< _IntType >::reset () [inline]

```

Resets the distribution state.

## t()

```

template<typename _IntType = int>
_IntType std::binomial_distribution< _IntType >::t () const [inline]

```

Returns the distribution t parameter.

## 5.256.4 Friends And Related Symbol Documentation

### operator<<

```

template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::binomial_distribution< _IntType1 > & __x) [friend]

```

Inserts a `binomial_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                |
| <code>__x</code>  | A <code>binomial_distribution</code> random number distribution. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _IntType = int>
bool operator== (
    const binomial\_distribution< _IntType > & __d1,
    const binomial\_distribution< _IntType > & __d2) [friend]
```

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic\_istream< _CharT, _Traits > & operator>> (
    std::basic\_istream< _CharT, _Traits > & __is,
    std::binomial\_distribution< _IntType1 > & __x) [friend]
```

Extracts a `binomial_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                     |
| <code>__x</code>  | A <code>binomial_distribution</code> random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.257 `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

```
#include <binomial_heap_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `base_type::allocator_type` **allocator\_type**
- typedef `base_type::cmp_fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **binomial\_heap** (const `binomial_heap` &)
- **binomial\_heap** (const `Cmp_Fn` &)
- **iterator** **begin** ()
- **const\_iterator** **begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator** **end** ()
- **const\_iterator** **end** () const
- void **erase** (`point_iterator`)
- template<typename `Pred`>  
  size\_type **erase\_if** (`Pred`)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- void **join** (`binomial_heap_base`< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- size\_type **max\_size** () const
- void **modify** (`point_iterator`, const\_reference)
- void **pop** ()
- `point_iterator` **push** (const\_reference)
- size\_type **size** () const
- template<typename `Pred`>  
  void **split** (`Pred`, `binomial_heap_base`< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- void **swap** (`left_child_next_sibling_heap`< `Value_Type`, `Cmp_Fn`, `_Alloc::size_type`, `_Alloc` > &)
- const\_reference **top** () const

### Protected Types

- typedef [base\\_type::node](#) **node**
- typedef `alloc_traits::allocator_type` **node\_allocator**
- typedef `_Alloc::size_type` **node\_metadata**
- typedef [std::pair](#)< `node_pointer`, `node_pointer` > **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (`node_pointer`)
- void **bubble\_to\_top** (`node_pointer`)
- void **clear\_imp** (`node_pointer`)
- template<typename `It`>  
void **copy\_from\_range** (`It`, `It`)
- void **find\_max** ()
- `node_pointer` **get\_new\_node\_for\_insert** (`const_reference`)
- `node_pointer` **prune** (`Pred`)
- void **swap** ([binomial\\_heap\\_base](#)< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- void **swap\_with\_parent** (`node_pointer`, `node_pointer`)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (`node_pointer`, `node_pointer`)
- static `node_pointer` **parent** (`node_pointer`)

### Protected Attributes

- `node_pointer` **m\_p\_max**
- `node_pointer` **m\_p\_root**
- `size_type` **m\_size**

#### 5.257.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>  
class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Binomial heap.

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_.hpp](#)

#### 5.258 `__gnu_pbds::detail::binomial_heap_base`< `Value_Type`, `Cmp_Fn`, `_Alloc` > Class Template Reference

```
#include <binomial_heap_base_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- template<typename Pred>  
  size\_type **erase\_if** (Pred)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- void **join** (`binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >` &)

- size\_type **max\_size** () const
- void **modify** ([point\\_iterator](#), const\_reference)
- void **pop** ()
- [point\\_iterator](#) **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, [binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** ([left\\_child\\_next\\_sibling\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference **top** () const

### Protected Types

- typedef [base\\_type::node](#) **node**
- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef base\_type::node\_const\_pointer **node\_const\_pointer**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef base\_type::node\_pointer **node\_pointer**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- **binomial\_heap\_base** (const [binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- **binomial\_heap\_base** (const Cmp\_Fn &)
- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It>  
void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 5.258.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >
```

Base class for binomial heap.

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_base\\_.hpp](#)

## 5.259 \_\_gnu\_pbds::binomial\_heap\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::binomial\_heap\_tag:



### 5.259.1 Detailed Description

Binomial-heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.260 \_\_gnu\_cxx::bitmap\_allocator< \_Tp > Class Template Reference

```
#include <bitmap_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::bitmap\_allocator< \_Tp >:





## Public Types

- typedef free\_list::\_\_mutex\_type **\_\_mutex\_type**
- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **bitmap\_allocator** (const [bitmap\\_allocator](#) &) noexcept
- template<typename \_Tp1>  
**bitmap\_allocator** (const [bitmap\\_allocator](#)< \_Tp1 > &) noexcept
- pointer **\_M\_allocate\_single\_object** ()
- void **\_M\_deallocate\_single\_object** (pointer \_\_p) throw ()
- const\_pointer **address** (const\_reference \_\_r) const noexcept
- pointer **address** (reference \_\_r) const noexcept
- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, typename [bitmap\\_allocator](#)< void >::const\_pointer)
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \* \_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n) throw ()
- template<typename \_Up>  
void **destroy** (\_Up \* \_\_p)
- size\_type **max\_size** () const noexcept

### 5.260.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::bitmap_allocator< _Tp >
```

Bitmap Allocator, primary template.

### 5.260.2 Member Function Documentation

#### **\_M\_allocate\_single\_object()**

```
template<typename _Tp>
pointer __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object () [inline]
Allocates memory for a single object of size sizeof(_Tp).
```

#### Exceptions

|                        |                                |
|------------------------|--------------------------------|
| <i>std::bad_alloc.</i> | If memory cannot be allocated. |
|------------------------|--------------------------------|

Complexity: Worst case complexity is O(N), but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of O(1)! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

**\_M\_deallocate\_single\_object()**

```
template<typename _Tp>
void __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object (
    pointer __p) throw ( ) [inline]
```

Deallocates memory that belongs to a single object of size sizeof(\_Tp).

Complexity: O(lg(N)), but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in O(1) time by the deallocate function.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

**5.261 std::\_\_debug::bitset< \_Nb > Class Template Reference**

```
#include <bitset>
```

**Public Types**

- typedef \_Base::reference **reference**

**Public Member Functions**

- constexpr **bitset** (const [\\_Base](#) &\_\_x)
- template<typename \_CharT, typename = \_Require<is\_trivially\_copyable<\_CharT>, is\_standard\_layout<\_CharT>, is\_trivially\_default\_constructible<\_CharT>, \_not\_<is\_array<\_CharT>>>>
 constexpr **bitset** (const \_CharT \*\_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<class \_CharT, class \_Traits, class \_Alloc>
 constexpr **bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc>
 constexpr **bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos=0, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n=([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::npos))
- constexpr **bitset** (unsigned long long \_\_val) noexcept
- constexpr const [\\_Base](#) & **\_M\_base** () const noexcept
- constexpr [\\_Base](#) & **\_M\_base** () noexcept
- constexpr [bitset](#)< \_Nb > & **flip** () noexcept
- constexpr [bitset](#)< \_Nb > & **flip** (size\_t \_\_pos)
- constexpr [bitset](#)< \_Nb > & **operator&=** (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- constexpr [bitset](#)< \_Nb > **operator<<** (size\_t \_\_pos) const noexcept
- constexpr [bitset](#)< \_Nb > & **operator<=** (size\_t \_\_pos) noexcept
- constexpr bool **operator==** (const [bitset](#)< \_Nb > &\_\_rhs) const noexcept
- constexpr [bitset](#)< \_Nb > **operator>>** (size\_t \_\_pos) const noexcept
- constexpr [bitset](#)< \_Nb > & **operator>=** (size\_t \_\_pos) noexcept
- constexpr reference **operator[]** (size\_t \_\_pos)
- constexpr bool **operator[]** (size\_t \_\_pos) const
- constexpr [bitset](#)< \_Nb > & **operator^=** (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- constexpr [bitset](#)< \_Nb > & **operator|=** (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- constexpr [bitset](#)< \_Nb > **operator~** () const noexcept
- constexpr [bitset](#)< \_Nb > & **reset** () noexcept
- constexpr [bitset](#)< \_Nb > & **reset** (size\_t \_\_pos)
- constexpr [bitset](#)< \_Nb > & **set** () noexcept

- constexpr [bitset](#)< \_Nb > & [set](#) (size\_t \_\_pos, bool \_\_val=true)
- template<typename \_CharT, typename \_Traits, typename \_Alloc>  
constexpr [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) () const
- template<typename \_CharT, typename \_Traits>  
constexpr [std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > [to\\_string](#) () const
- template<typename \_CharT>  
constexpr [std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > [to\\_string](#) () const
- constexpr [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > [to\\_string](#) () const
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<class \_CharT, class \_Traits>  
constexpr [std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<class \_CharT>  
constexpr [std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- constexpr [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > [to\\_string](#) (char \_\_zero, char \_\_one='1') const

### 5.261.1 Detailed Description

```
template<size_t _Nb>
class std::__debug::bitset< _Nb >
```

Class std::bitset with additional safety/checking/debug instrumentation.  
The documentation for this class was generated from the following file:

- [debug/bitset](#)

## 5.262 std::bitset< \_Nb > Class Template Reference

```
#include <bitset>
```

Inheritance diagram for std::bitset< \_Nb >:



### Classes

- class [reference](#)

## Public Member Functions

- constexpr [bitset](#) () noexcept
- template<typename \_CharT, typename = \_Require<is\_trivially\_copyable<\_CharT>, is\_standard\_layout<\_CharT>, is\_trivially\_default\_constructible<\_CharT>, \_not\_<is\_array<\_CharT>>>>  
constexpr [bitset](#) (const \_CharT \* \_\_str, typename [\\_\\_bitset::\\_\\_string](#)<\_CharT>::size\_type \_\_n= [\\_\\_bitset::\\_\\_string](#)<\_CharT>::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [bitset](#) (const [std::basic\\_string](#)<\_CharT, \_Traits, \_Alloc> & \_\_s, size\_t \_\_position, size\_t \_\_n)
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [bitset](#) (const [std::basic\\_string](#)<\_CharT, \_Traits, \_Alloc> & \_\_s, size\_t \_\_position, size\_t \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [bitset](#) (const [std::basic\\_string](#)<\_CharT, \_Traits, \_Alloc> & \_\_s, size\_t \_\_position=0)
- constexpr [bitset](#) (unsigned long long \_\_val) noexcept
- constexpr size\_t [Find\\_first](#) () const noexcept
- constexpr size\_t [Find\\_next](#) (size\_t \_\_prev) const noexcept
- constexpr bool [all](#) () const noexcept
- constexpr bool [any](#) () const noexcept
- constexpr size\_t [count](#) () const noexcept
- constexpr [bitset](#)<\_Nb> & [flip](#) () noexcept
- constexpr [bitset](#)<\_Nb> & [flip](#) (size\_t \_\_position)
- constexpr bool [none](#) () const noexcept
- constexpr [bitset](#)<\_Nb> [operator~](#) () const noexcept
- constexpr [bitset](#)<\_Nb> & [reset](#) () noexcept
- constexpr [bitset](#)<\_Nb> & [reset](#) (size\_t \_\_position)
- constexpr [bitset](#)<\_Nb> & [set](#) () noexcept
- constexpr [bitset](#)<\_Nb> & [set](#) (size\_t \_\_position, bool \_\_val=true)
- constexpr size\_t [size](#) () const noexcept
- constexpr bool [test](#) (size\_t \_\_position) const
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [std::basic\\_string](#)<\_CharT, \_Traits, \_Alloc> [to\\_string](#) () const
- template<class \_CharT, class \_Traits>  
constexpr [std::basic\\_string](#)<\_CharT, \_Traits, [std::allocator](#)<\_CharT>> [to\\_string](#) () const
- template<class \_CharT>  
constexpr [std::basic\\_string](#)<\_CharT, [std::char\\_traits](#)<\_CharT>, [std::allocator](#)<\_CharT>> [to\\_string](#) () const
- constexpr [std::basic\\_string](#)<char, [std::char\\_traits](#)<char>, [std::allocator](#)<char>> [to\\_string](#) () const
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [std::basic\\_string](#)<\_CharT, \_Traits, \_Alloc> [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<class \_CharT, class \_Traits>  
constexpr [std::basic\\_string](#)<\_CharT, \_Traits, [std::allocator](#)<\_CharT>> [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<class \_CharT>  
constexpr [std::basic\\_string](#)<\_CharT, [std::char\\_traits](#)<\_CharT>, [std::allocator](#)<\_CharT>> [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- constexpr [std::basic\\_string](#)<char, [std::char\\_traits](#)<char>, [std::allocator](#)<char>> [to\\_string](#) (char \_\_zero, char \_\_one='1') const
- constexpr unsigned long long [to\\_ullong](#) () const
- constexpr unsigned long [to\\_ulong](#) () const
- constexpr [bitset](#)<\_Nb> & [operator&=](#) (const [bitset](#)<\_Nb> & \_\_rhs) noexcept

- constexpr `bitset<_Nb> & operator|=` (const `bitset<_Nb> &__rhs`) noexcept
- constexpr `bitset<_Nb> & operator^=` (const `bitset<_Nb> &__rhs`) noexcept
- constexpr `bitset<_Nb> & operator<<=` (size\_t \_\_position) noexcept
- constexpr `bitset<_Nb> & operator>>=` (size\_t \_\_position) noexcept
- constexpr `bitset<_Nb> & _Unchecked_set` (size\_t \_\_pos) noexcept
- constexpr `bitset<_Nb> & _Unchecked_set` (size\_t \_\_pos, int \_\_val) noexcept
- constexpr `bitset<_Nb> & _Unchecked_reset` (size\_t \_\_pos) noexcept
- constexpr `bitset<_Nb> & _Unchecked_flip` (size\_t \_\_pos) noexcept
- constexpr bool `_Unchecked_test` (size\_t \_\_pos) const noexcept
- constexpr `reference operator[]` (size\_t \_\_position)
- constexpr bool `operator[]` (size\_t \_\_position) const
- constexpr bool `operator==` (const `bitset<_Nb> &__rhs`) const noexcept
- constexpr `bitset<_Nb> operator<<` (size\_t \_\_position) const noexcept
- constexpr `bitset<_Nb> operator>>` (size\_t \_\_position) const noexcept

## Friends

- template<class \_CharT, class \_Traits, size\_t \_Nb2>  
`std::basic_ostream<_CharT, _Traits> & operator<<` (`std::basic_ostream<_CharT, _Traits> &`, const `bitset<_Nb2> &`)
- template<class \_CharT, class \_Traits, size\_t \_Nb2>  
`std::basic_istream<_CharT, _Traits> & operator>>` (`std::basic_istream<_CharT, _Traits> &`, `bitset<_Nb2> &`)
- class **reference**
- struct `std::hash<bitset>`

## 5.262.1 Detailed Description

**template<size\_t \_Nb>**  
**class std::bitset<\_Nb>**

The `bitset` class represents a *fixed-size* sequence of bits.

(Note that `bitset` does *not* meet the formal requirements of a [container](#). Mainly, it lacks iterators.)

The template argument, `Nb`, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024\*1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let `B` be the number of bits in a word, then  $(Nb + (B - 1)) / B$  words will be used for storage. `B - Nb % B` bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `bitset` as a *simple array of bits*, be aware that your mental picture is reversed: a `bitset` behaves the same way as bits in integers do, with the bit at index 0 in the *least significant / right-hand* position, and the bit at index `Nb-1` in the *most significant / left-hand* position. Thus, unlike other containers, a `bitset`'s index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long      a = 'a';
    bitset<10> b(a);

    cout << "b('a') is " << b << endl;

    ostringstream s;
    s << b;
    string str = s.str();
    cout << "index 3 in the string is " << str[3] << " but\n"
         << "index 3 in the bitset is " << b[3] << endl;
}
```

Also see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext\\_containers.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_containers.html) for a description of extensions.

Most of the actual code isn't contained in `bitset<>` itself, but in the base class `_Base_bitset`. The base class works with whole words, not with individual bits. This allows us to specialize `_Base_bitset` for the important special case where the bitset is only a single word.

Extra confusion can result due to the fact that the storage for `_Base_bitset` is a regular array, and is indexed as such. This is carefully encapsulated.

### 5.262.2 Constructor & Destructor Documentation

#### bitset() [1/5]

```
template<size_t _Nb>
std::bitset<_Nb>::bitset () [inline], [constexpr], [noexcept]
All bits set to zero.
```

#### bitset() [2/5]

```
template<size_t _Nb>
std::bitset<_Nb>::bitset (
    unsigned long long __val) [inline], [constexpr], [noexcept]
Initial bits bitwise-copied from a single word (others set to zero).
```

#### bitset() [3/5]

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::bitset<_Nb>::bitset (
    const std::basic_string<_CharT, _Traits, _Alloc> & __s,
    size_t __position = 0) [inline], [explicit], [constexpr]
```

Use a subset of a string.

#### Parameters

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <code>__s</code>        | A string of 0 and 1 characters.                                            |
| <code>__position</code> | Index of the first character in <code>__s</code> to use; defaults to zero. |

## Exceptions

|                              |                                                                |
|------------------------------|----------------------------------------------------------------|
| <i>std::out_of_range</i>     | If <code>__position &gt; __s.size()</code> .                   |
| <i>std::invalid_argument</i> | If a character appears in the string which is neither 0 nor 1. |

**bitset()** [4/5]

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::bitset<_Nb>::bitset (
    const std::basic_string<_CharT, _Traits, _Alloc> & __s,
    size_t __position,
    size_t __n) [inline], [constexpr]
```

Use a subset of a string.

## Parameters

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>__s</code>        | A string of 0 and 1 characters.                          |
| <code>__position</code> | Index of the first character in <code>__s</code> to use. |
| <code>__n</code>        | The number of characters to copy.                        |

## Exceptions

|                              |                                                                |
|------------------------------|----------------------------------------------------------------|
| <i>std::out_of_range</i>     | If <code>__position &gt; __s.size()</code> .                   |
| <i>std::invalid_argument</i> | If a character appears in the string which is neither 0 nor 1. |

**bitset()** [5/5]

```
template<size_t _Nb>
template<typename _CharT, typename = _Require<is_trivially_copyable<_CharT>, is_standard_↵
layout<_CharT>, is_trivially_default_constructible<_CharT>, __not<is_array<_CharT>>>>
std::bitset<_Nb>::bitset (
    const _CharT * __str,
    typename __bitset<_Nb>::__string<_CharT>::size_type __n = __bitset<_Nb>::__↵
string<_CharT>::npos,
    _CharT __zero = _CharT('0'),
    _CharT __one = _CharT('1')) [inline], [explicit], [constexpr]
```

Construct from a character array.

## Parameters

|                     |                                                                     |
|---------------------|---------------------------------------------------------------------|
| <code>__str</code>  | An array of characters <code>__zero</code> and <code>__one</code> . |
| <code>__n</code>    | The number of characters to use.                                    |
| <code>__zero</code> | The character corresponding to the value 0.                         |
| <code>__one</code>  | The character corresponding to the value 1.                         |

**Exceptions**

|                                    |                                                                                                    |
|------------------------------------|----------------------------------------------------------------------------------------------------|
| <code>std::invalid_argument</code> | If a character appears in the string which is neither <code>__zero</code> nor <code>__one</code> . |
|------------------------------------|----------------------------------------------------------------------------------------------------|

**5.262.3 Member Function Documentation****all()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::all () const [inline], [constexpr], [noexcept]
```

Tests whether all the bits are on.

**Returns**

True if all the bits are set.

**any()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::any () const [inline], [constexpr], [noexcept]
```

Tests whether any of the bits are on.

**Returns**

True if at least one bit is set.

**count()**

```
template<size_t _Nb>
size_t std::bitset<_Nb>::count () const [inline], [constexpr], [noexcept]
```

Returns the number of bits which are set.

**flip() [1/2]**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::flip () [inline], [constexpr], [noexcept]
```

Toggles every bit to its opposite value.

**flip() [2/2]**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::flip (
    size_t __position) [inline], [constexpr]
```

Toggles a given bit to its opposite value.

**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>__position</code> | The index of the bit. |
|-------------------------|-----------------------|

**Exceptions**

|                                |                                                    |
|--------------------------------|----------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is bigger the size of the set. |
|--------------------------------|----------------------------------------------------|



**none()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::none () const [inline], [constexpr], [noexcept]
```

Tests whether any of the bits are on.

**Returns**

True if none of the bits are set.

**operator&=()**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::operator&= (
    const bitset<_Nb> & __rhs) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__rhs</code> | A same-sized bitset. |
|--------------------|----------------------|

These should be self-explanatory.

**operator<<()**

```
template<size_t _Nb>
bitset<_Nb> > std::bitset<_Nb>::operator<< (
    size_t __position) const [inline], [constexpr], [noexcept]
```

Self-explanatory.

**operator<<=()**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::operator<<= (
    size_t __position) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

**Parameters**

|                         |                                |
|-------------------------|--------------------------------|
| <code>__position</code> | The number of places to shift. |
|-------------------------|--------------------------------|

These should be self-explanatory.

**operator==()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::operator== (
    const bitset<_Nb> & __rhs) const [inline], [constexpr], [noexcept]
```

These comparisons for equality/inequality are, well, *bitwise*.

**operator>>()**

```
template<size_t _Nb>
bitset<_Nb> > std::bitset<_Nb>::operator>> (
    size_t __position) const [inline], [constexpr], [noexcept]
```

Self-explanatory.

**operator>>=()**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::operator>>= (
    size_t __position) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

**Parameters**

|                         |                                |
|-------------------------|--------------------------------|
| <code>__position</code> | The number of places to shift. |
|-------------------------|--------------------------------|

These should be self-explanatory.

**operator[]() [1/2]**

```
template<size_t _Nb>
reference std::bitset<_Nb>::operator[] (
    size_t __position) [inline], [constexpr]
```

Array-indexing support.

**Parameters**

|                         |                        |
|-------------------------|------------------------|
| <code>__position</code> | Index into the bitset. |
|-------------------------|------------------------|

**Returns**

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

**operator[]() [2/2]**

```
template<size_t _Nb>
bool std::bitset<_Nb>::operator[] (
    size_t __position) const [inline], [constexpr]
```

Array-indexing support.

**Parameters**

|                         |                        |
|-------------------------|------------------------|
| <code>__position</code> | Index into the bitset. |
|-------------------------|------------------------|

**Returns**

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

**operator^=()**

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator^= (
    const bitset< _Nb > & __rhs) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__rhs</code> | A same-sized bitset. |
|--------------------|----------------------|

These should be self-explanatory.

**operator" |=()**

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator|= (
    const bitset< _Nb > & __rhs) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__rhs</code> | A same-sized bitset. |
|--------------------|----------------------|

These should be self-explanatory.

**operator~()**

```
template<size_t _Nb>
bitset< _Nb > std::bitset< _Nb >::operator~ () const [inline], [constexpr], [noexcept]
```

See the no-argument flip().

**reset() [1/2]**

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::reset () [inline], [constexpr], [noexcept]
```

Sets every bit to false.

**reset() [2/2]**

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::reset (
    size_t __position) [inline], [constexpr]
```

Sets a given bit to false.

**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>__position</code> | The index of the bit. |
|-------------------------|-----------------------|

**Exceptions**

|                                |                                              |
|--------------------------------|----------------------------------------------|
| <code>std::out_of_range</code> | If <i>pos</i> is bigger the size of the set. |
|--------------------------------|----------------------------------------------|

Same as writing `set (pos, false)`.

**set()** [1/2]

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::set () [inline], [constexpr], [noexcept]
```

Sets every bit to true.

**set()** [2/2]

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::set (
    size_t __position,
    bool __val = true) [inline], [constexpr]
```

Sets a given bit to a particular value.

**Parameters**

|                         |                                         |
|-------------------------|-----------------------------------------|
| <code>__position</code> | The index of the bit.                   |
| <code>__val</code>      | Either true or false, defaults to true. |

**Exceptions**

|                                |                                              |
|--------------------------------|----------------------------------------------|
| <code>std::out_of_range</code> | If <i>pos</i> is bigger the size of the set. |
|--------------------------------|----------------------------------------------|

**size()**

```
template<size_t _Nb>
size_t std::bitset<_Nb>::size () const [inline], [constexpr], [noexcept]
```

Returns the total number of bits.

**test()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::test (
    size_t __position) const [inline], [constexpr]
```

Tests the value of a bit.

**Parameters**

|                         |                     |
|-------------------------|---------------------|
| <code>__position</code> | The index of a bit. |
|-------------------------|---------------------|

**Returns**

The value at *pos*.

**Exceptions**

|                                |                                              |
|--------------------------------|----------------------------------------------|
| <code>std::out_of_range</code> | If <i>pos</i> is bigger the size of the set. |
|--------------------------------|----------------------------------------------|

**to\_string()**

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::basic_string< _CharT, _Traits, _Alloc > std::bitset< _Nb >::to_string () const [inline],
[constexpr]
```

Returns a character interpretation of the bitset.

**Returns**

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

**to\_ulong()**

```
template<size_t _Nb>
unsigned long std::bitset< _Nb >::to_ulong () const [inline], [constexpr]
```

Returns a numerical interpretation of the bitset.

**Returns**

The integral equivalent of the bits.

**Exceptions**

|                                  |                                                                   |
|----------------------------------|-------------------------------------------------------------------|
| <code>std::overflow_error</code> | If there are too many bits to be represented in an unsigned long. |
|----------------------------------|-------------------------------------------------------------------|

The documentation for this class was generated from the following file:

- [bitset](#)

**5.263 std::tr2::bool\_set Class Reference**

```
#include <bool_set>
```

**Public Member Functions**

- constexpr [bool\\_set](#) ()
- constexpr [bool\\_set](#) (bool \_\_t)
- bool **contains** ([bool\\_set](#) \_\_b) const
- bool **equals** ([bool\\_set](#) \_\_b) const
- bool **is\_emptyset** () const
- bool **is\_indeterminate** () const
- bool **is\_singleton** () const
- **operator bool** () const

**Static Public Member Functions**

- static [bool\\_set](#) **emptyset** ()
- static [bool\\_set](#) **indeterminate** ()

## Friends

- [bool\\_set operator!](#) ([bool\\_set](#) \_\_b)
- [bool\\_set operator&](#) ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- [template<typename CharT, typename Traits>](#)  
[std::basic\\_ostream](#)< CharT, Traits > & [operator](#)<< ([std::basic\\_ostream](#)< CharT, Traits > &\_\_out, [bool\\_set](#) \_\_b)
- [bool\\_set operator==](#) ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- [template<typename CharT, typename Traits>](#)  
[std::basic\\_istream](#)< CharT, Traits > & [operator](#)>> ([std::basic\\_istream](#)< CharT, Traits > &\_\_in, [bool\\_set](#) &\_\_b)
- [bool\\_set operator^](#) ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set operator|](#) ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)

### 5.263.1 Detailed Description

[bool\\_set](#)

See N2136, [Bool\\_set](#): multi-valued logic by Hervé Brönnimann, Guillaume Melquiond, Sylvain Pion.

The implicit conversion to bool is slippery! I may use the new explicit conversion. This has been specialized in the language so that in contexts requiring a bool the conversion happens implicitly. Thus most objections should be eliminated.

### 5.263.2 Constructor & Destructor Documentation

#### [bool\\_set\(\)](#) [1/2]

```
std::tr2::bool_set::bool_set () [inline], [constexpr]
```

Default constructor.

#### [bool\\_set\(\)](#) [2/2]

```
std::tr2::bool_set::bool_set (
    bool __t) [inline], [constexpr]
```

Constructor from bool.

### 5.263.3 Member Function Documentation

#### [equals\(\)](#)

```
bool std::tr2::bool_set::equals (
    bool_set __b) const [inline]
```

Return true if states are equal.

#### [is\\_emptyset\(\)](#)

```
bool std::tr2::bool_set::is_emptyset () const [inline]
```

Return true if this is empty.

#### [is\\_indeterminate\(\)](#)

```
bool std::tr2::bool_set::is_indeterminate () const [inline]
```

Return true if this is indeterminate.

#### [is\\_singleton\(\)](#)

```
bool std::tr2::bool_set::is_singleton () const [inline]
```

Return true if this is false or true (normal boolean).

## operator bool()

```
std::tr2::bool_set::operator bool () const [inline]
```

Conversion to bool.

The documentation for this class was generated from the following files:

- [bool\\_set](#)
- [bool\\_set.tcc](#)

## 5.264 \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Itr, \_Alloc > Struct Template Reference

```
#include <branch_policy.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Itr, \_Alloc >:



## Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `rebind_k::const_reference` **key\_const\_reference**
- typedef `value_type::first_type` **key\_type**
- typedef `remove_const< key_type >::type` **rckey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `rebind_traits< _Alloc, rckey_type >` **rebind\_k**
- typedef `rebind_traits< _Alloc, rcvalue_type >` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

## Protected Member Functions

- virtual `it_type end ()=0`
- `it_type end_iterator () const`

## Static Protected Member Functions

- static `key_const_reference extract_key (const_reference r_val)`

### 5.264.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >
```

Primary template, base class for branch structure policies.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

## 5.265 `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >` Struct Template Reference

```
#include <branch_policy.hpp>
```

### Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Cltr::value_type` **it\_type**
- typedef `Node_Cltr::value_type` **it\_type**
- typedef `rebind_k::const_reference` **key\_const\_reference**
- typedef `rebind_v::const_reference` **key\_const\_reference**
- typedef `value_type::first_type` **key\_type**
- typedef `value_type` **key\_type**
- typedef `remove_const< key_type >::type` **rkey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef [rebind\\_traits](#)< `_Alloc`, `rkey_type` > **rebind\_k**
- typedef [rebind\\_traits](#)< `_Alloc`, `rcvalue_type` > **rebind\_v**
- typedef [rebind\\_traits](#)< `_Alloc`, `rcvalue_type` > **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `rebind_v::reference` **reference**
- typedef [std::iterator\\_traits](#)< `it_type` >::`value_type` **value\_type**
- typedef [std::iterator\\_traits](#)< `it_type` >::`value_type` **value\_type**

### Protected Member Functions

- virtual `it_type` **end** () const =0
- virtual `it_type` **end** ()=0
- `it_type` **end\_iterator** () const
- `it_type` **end\_iterator** () const

### Static Protected Member Functions

- static `key_const_reference` **extract\_key** (const\_reference r\_val)
- static `key_const_reference` **extract\_key** (const\_reference r\_val)



### 5.265.1 Detailed Description

```
template<typename Node_Cltr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >
```

Specialization for const iterators.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

## 5.266 std::cauchy\_distribution<\_RealType> Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **cauchy\_distribution** (\_RealType \_\_a, \_RealType \_\_b=1.0)
- **cauchy\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

### Friends

- bool **operator==** (const [cauchy\\_distribution](#) &\_\_d1, const [cauchy\\_distribution](#) &\_\_d2)

### 5.266.1 Detailed Description

```
template<typename _RealType = double>
class std::cauchy_distribution< _RealType >
```

A cauchy\_distribution random number distribution.

The formula for the normal probability mass function is  $p(x|a,b) = (\pi b(1 + (\frac{x-a}{b})^2))^{-1}$

Since

C++11

### 5.266.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::cauchy_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 5.266.3 Member Function Documentation

#### max()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::cauchy_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

#### param() [1/2]

```
template<typename _RealType = double>
param_type std::cauchy_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

#### param() [2/2]

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

#### reset()

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::reset () [inline]
Resets the distribution state.
```

### 5.266.4 Friends And Related Symbol Documentation

#### operator==

```
template<typename _RealType = double>
bool operator== (
    const cauchy_distribution< _RealType > & __d1,
    const cauchy_distribution< _RealType > & __d2) [friend]
```

Return true if two Cauchy distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 5.267 `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

```
#include <hash_policy.hpp>
```

#### Public Types

- enum { [external\\_load\\_access](#) }
- typedef Size\_Type **size\_type**

#### Public Member Functions

- [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#) (float load=0.5)
- float [get\\_load](#) () const
- void [set\\_load](#) (float load)
- void **swap** ([cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)< External\_Load\_Access, Size\_Type > &other)

#### Protected Member Functions

- bool [is\\_grow\\_needed](#) (size\_type size, size\_type num\_entries) const
- bool [is\\_resize\\_needed](#) () const
- void [notify\\_cleared](#) ()
- void [notify\\_erase\\_search\\_collision](#) ()
- void [notify\\_erase\\_search\\_end](#) ()
- void [notify\\_erase\\_search\\_start](#) ()
- void [notify\\_erased](#) (size\_type num\_entries)
- void [notify\\_externally\\_resized](#) (size\_type new\_size)
- void [notify\\_find\\_search\\_collision](#) ()
- void [notify\\_find\\_search\\_end](#) ()

- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (size\_type num\_entries)
- void `notify_resized` (size\_type new\_size)

### 5.267.1 Detailed Description

`template<bool External_Load_Access = false, typename Size_Type = std::size_t>`  
`class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >`

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.

### 5.267.2 Member Enumeration Documentation

#### anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

#### Enumerator

|                                   |                                                                                                                                                                 |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>external_load_access</code> | Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation. |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|

### 5.267.3 Constructor & Destructor Documentation

#### `cc_hash_max_collision_check_resize_trigger()`

```
template<bool External_Load_Access, typename Size_Type>
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_↵
hash_max_collision_check_resize_trigger (
    float load = 0.5)
```

Default constructor, or constructor taking load, a `__load` factor which it will attempt to maintain.

### 5.267.4 Member Function Documentation

#### `get_load()`

```
template<bool External_Load_Access, typename Size_Type>
float __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::get_load () const [inline]
```

Returns the current load.

#### `is_grow_needed()`

```
template<bool External_Load_Access, typename Size_Type>
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::is_grow_needed (
    size_type size,
    size_type num_entries) const [inline], [protected]
```

Queries whether a grow is needed. This method is called only if this object indicated is needed.

**is\_resize\_needed()**

```
template<bool External_Load_Access, typename Size_Type>
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::is_resize_needed () const [inline], [protected]
```

Queries whether a resize is needed.

**notify\_cleared()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_cleared () [protected]
```

Notifies the table was cleared.

References [notify\\_resized\(\)](#).

**notify\_erase\_search\_collision()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erase_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_erase\_search\_end()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erase_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_erase\_search\_start()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erase_search_start () [inline], [protected]
```

Notifies an erase search started.

**notify\_erased()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erased (
    size_type num_entries) [inline], [protected]
```

Notifies an element was erased.

**notify\_externally\_resized()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_externally_resized (
    size_type new_size) [protected]
```

Notifies the table was resized externally.

**notify\_find\_search\_collision()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_find\_search\_end()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_find\_search\_start()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_find_search_start () [inline], [protected]
```

Notifies a find search started.

**notify\_insert\_search\_collision()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_insert\_search\_end()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_insert\_search\_start()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_insert_search_start () [inline], [protected]
```

Notifies an insert search started.

**notify\_inserted()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_inserted (
    size_type num_entries) [inline], [protected]
```

Notifies an element was inserted.

**notify\_resized()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_resized (
    size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.  
Referenced by [notify\\_cleared\(\)](#).

### set\_load()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↔
::set_load (
    float load)
```

Sets the load; does not resize the container.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 5.268 \_\_gnu\_pbds::cc\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc > Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::cc\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >:



### Public Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn**
- typedef [cc\\_hash\\_tag](#) **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef Resize\_Policy **resize\_policy**

### Public Member Functions

- [cc\\_hash\\_table](#) ()
- **cc\_hash\_table** (const [cc\\_hash\\_table](#) &other)
- [cc\\_hash\\_table](#) (const hash\_fn &h)
- [cc\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e)
- [cc\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- [cc\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- template<typename It>  
  [cc\\_hash\\_table](#) (It first, It last)

- `template<typename It>`  
`cc_hash_table` (It first, It last, const hash\_fn &h)
- `template<typename It>`  
`cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e)
- `template<typename It>`  
`cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- `template<typename It>`  
`cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- `cc_hash_table` & **operator=** (const `cc_hash_table` &other)
- void **swap** (`cc_hash_table` &other)

### 5.268.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename _Alloc = std::allocator<char>>>
class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A collision-chaining hash-based associative container.

#### Template Parameters

|                      |                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                       |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                       |
| <i>Hash_Fn</i>       | Hashing functor.                                                                                                                                                                                |
| <i>Eq_Fn</i>         | Equal functor.                                                                                                                                                                                  |
| <i>Comb_Hash_Fn</i>  | Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) |
| <i>Resize_Policy</i> | Resizes hash.                                                                                                                                                                                   |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true                                         |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                 |

Base tag choices are: `cc_hash_tag`.

Base is `basic_hash_table`.

### 5.268.2 Constructor & Destructor Documentation

#### `cc_hash_table()` [1/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename _Alloc = std::allocator<char>>>
```

```
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table () [inline]
```

Default constructor.



**cc\_hash\_table()** [2/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    const hash_fn & h) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `Hash_Fn` object of the container object.

**cc\_hash\_table()** [3/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    const hash_fn & h,
    const eq_fn & e) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

**cc\_hash\_table()** [4/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_hash_fn & ch) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

**cc\_hash\_table()** [5/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_hash_fn & ch,
    const resize_policy & rp) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

#### `cc_hash_table()` [6/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type,
bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
    It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

#### `cc_hash_table()` [7/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type,
bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
    It last,
    const hash_fn & h) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

#### `cc_hash_table()` [8/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type,
bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

**cc\_hash\_table()** [9/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_hash_fn & ch) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_↵ it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, and r\_comb\_hash\_fn will be copied by the comb\_hash\_fn object of the container object.

**cc\_hash\_table()** [10/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_hash_fn & ch,
    const resize_policy & rp) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, r\_comb\_hash\_fn will be copied by the comb\_hash\_fn object of the container object, and r\_resize\_policy will be copied by the resize\_policy object of the container object.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

**5.269 \_\_gnu\_pbds::cc\_hash\_tag Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::cc_hash_tag`:



### 5.269.1 Detailed Description

Collision-chaining hash.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.270 `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >` Class Template Reference

```
#include <cc_ht_map_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_`

`_Hash_Fn, Resize_Policy >`:



## Public Types

- enum { **store\_hash** }
- typedef `_Alloc` **allocator\_type**
- typedef `Comb_Hash_Fn` **comb\_hash\_fn**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `point_const_iterator` **point\_const\_iterator**
- typedef `point_iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `Resize_Policy` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `traits_base::value_type` **value\_type**

## Public Member Functions

- `cc_ht_map` (const `cc_ht_map`< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)
- `cc_ht_map` (const Hash\_Fn &)
- `cc_ht_map` (const Hash\_Fn &, const Eq\_Fn &)
- `cc_ht_map` (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &)
- `cc_ht_map` (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &, const Resize\_Policy &)
- iterator `begin` ()
- const\_iterator `begin` () const
- void `clear` ()
- template<typename It>  
void `copy_from_range` (It, It)
- bool `empty` () const
- iterator `end` ()
- const\_iterator `end` () const
- bool `erase` (key\_const\_reference)
- template<typename Pred>  
size\_type `erase_if` (Pred)
- point\_iterator `find` (key\_const\_reference)
- point\_const\_iterator `find` (key\_const\_reference) const
- point\_iterator `find_end` ()
- point\_const\_iterator `find_end` () const
- Comb\_Hash\_Fn & `get_comb_hash_fn` ()
- const Comb\_Hash\_Fn & `get_comb_hash_fn` () const
- Eq\_Fn & `get_eq_fn` ()
- const Eq\_Fn & `get_eq_fn` () const
- Hash\_Fn & `get_hash_fn` ()
- const Hash\_Fn & `get_hash_fn` () const
- Resize\_Policy & `get_resize_policy` ()
- const Resize\_Policy & `get_resize_policy` () const
- void `initialize` ()
- `std::pair`< point\_iterator, bool > `insert` (const\_reference r\_val)
- size\_type `max_size` () const
- mapped\_reference `operator[]` (key\_const\_reference r\_key)
- size\_type `size` () const
- void `swap` (`cc_ht_map`< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)

## Public Attributes

- no\_throw\_indicator `m_no_throw_copies_indicator`
- store\_extra `m_store_extra_indicator`

## Friends

- class `const_iterator_`
- class `iterator_`

### 5.270.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn,
Resize_Policy >
```

A collision-chaining hash-based container.

#### Template Parameters

|                      |                                                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                                                                               |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                                                                               |
| <i>Hash_Fn</i>       | Hashing functor. Default is <code>__gnu_cxx::hash</code> .                                                                                                                                                                                              |
| <i>Eq_Fn</i>         | Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>                                                                                                                                                                                            |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                                                                         |
| <i>Store_Hash</i>    | If key type stores extra metadata. Defaults to false.                                                                                                                                                                                                   |
| <i>Comb_Hash_Fn</i>  | Combining hash functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) Default <code>direct_mask_range_hashing</code> . |
| <i>Resize_Policy</i> | Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .                                                                                  |

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_hash_fn`, `detail::types_traits`. (Optional: `detail::debug_map_base`.)

### 5.270.2 Member Enumeration Documentation

#### anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
```

anonymous enum

Value stores hash, true or false.

### 5.270.3 Member Function Documentation

#### empty()

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_
Fn, Resize_Policy >::empty () const [inline], [nodiscard]
True if size() == 0.
```

#### get\_comb\_hash\_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ()
Return current comb_hash_fn.
```

### `get_comb_hash_fn()` [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn () const
Return current const comb_hash_fn.
```

### `get_eq_fn()` [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Hash_Fn, Resize_Policy >::get_eq_fn ()
Return current eq_fn.
```

### `get_eq_fn()` [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_eq_fn () const
Return current const eq_fn.
```

### `get_hash_fn()` [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Hash_Fn, Resize_Policy >::get_hash_fn ()
Return current hash_fn.
```

### `get_hash_fn()` [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_hash_fn () const
Return current const hash_fn.
```

### `get_resize_policy()` [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_resize_policy ()
Return current resize_policy.
```

### `get_resize_policy()` [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy () const
Return current const resize_policy.
```

The documentation for this class was generated from the following file:



- [cc\\_ht\\_map.hpp](#)

## 5.271 `__gnu_cxx::char_traits<_CharT>` Struct Template Reference

```
#include <char_traits.h>
```

Inheritance diagram for `__gnu_cxx::char_traits<_CharT>`:



### Public Types

- typedef `_CharT` **char\_type**
- using **comparison\_category**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `_Char_types<_CharT>::pos_type` **pos\_type**
- typedef `_Char_types<_CharT>::state_type` **state\_type**

### Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \*\_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

### 5.271.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::char_traits<_CharT>
```

Base class used to implement std::char\_traits.

#### Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the `int_type` and `state_type` typedefs, and the `eof()` member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace `std` may not be specialized for fundamental types, but classes in namespace `__gnu_cxx` may be.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character\\_types](https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types) for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.272 std::char\_traits<\_CharT> Struct Template Reference

```
#include <char_traits.h>
```

Inheritance diagram for std::char\_traits<\_CharT>:



### Public Types

- typedef `_CharT` **char\_type**
- using **comparison\_category**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `_Char_types<_CharT>::pos_type` **pos\_type**
- typedef `_Char_types<_CharT>::state_type` **state\_type**

## Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

### 5.272.1 Detailed Description

```
template<typename _CharT>
struct std::char_traits< _CharT >
```

Basis for explicit traits specializations.

#### Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around `__gnu_cxx::char_traits`, it is possible to achieve a more appropriate definition by specializing `__gnu_cxx::char_traits`.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character\\_types](https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types) for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.273 std::char\_traits< \_\_gnu\_cxx::character< \_Value, \_Int, \_St > > Struct Template Reference

```
#include <pod_char_traits.h>
```

Inheritance diagram for std::char\_traits< \_\_gnu\_cxx::character< \_Value, \_Int, \_St > >:



### Public Types

- typedef `__gnu_cxx::character< _Value, _Int, _St >` **char\_type**
- typedef `__gnu_cxx::character< _Value, _Int, _St >` **char\_type**
- using **comparison\_category**
- using **comparison\_category**
- typedef `_Char_types< __gnu_cxx::character< _Value, _Int, _St > >::int_type` **int\_type**
- typedef `char_type::int_type` **int\_type**
- typedef `_Char_types< __gnu_cxx::character< _Value, _Int, _St > >::off_type` **off\_type**
- typedef `streamoff` **off\_type**
- typedef `_Char_types< __gnu_cxx::character< _Value, _Int, _St > >::pos_type` **pos\_type**
- typedef `fpos< state_type >` **pos\_type**
- typedef `_Char_types< __gnu_cxx::character< _Value, _Int, _St > >::state_type` **state\_type**
- typedef `char_type::state_type` **state\_type**

### Static Public Member Functions

- static constexpr void **assign** (`char_type &__c1`, const `char_type &__c2`)
- static constexpr `char_type * assign` (`char_type * __s`, std::size\_t \_\_n, `char_type __a`)
- static constexpr void **assign** (`char_type &__c1`, const `char_type &__c2`)
- static void **assign** (`char_type &__c1`, const `char_type &__c2`)
- static `char_type * assign` (`char_type * __s`, size\_t \_\_n, `char_type __a`)
- static constexpr `char_type * assign` (`char_type * __s`, std::size\_t \_\_n, `char_type __a`)
- static constexpr int **compare** (const `char_type * __s1`, const `char_type * __s2`, std::size\_t \_\_n)
- static int **compare** (const `char_type * __s1`, const `char_type * __s2`, size\_t \_\_n)
- static constexpr int **compare** (const `char_type * __s1`, const `char_type * __s2`, std::size\_t \_\_n)
- static constexpr `char_type * copy` (`char_type * __s1`, const `char_type * __s2`, std::size\_t \_\_n)
- static `char_type * copy` (`char_type * __s1`, const `char_type * __s2`, size\_t \_\_n)
- static constexpr `char_type * copy` (`char_type * __s1`, const `char_type * __s2`, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static int\_type **eof** ()
- static constexpr bool **eq** (const `char_type &__c1`, const `char_type &__c2`)

- static constexpr bool **eq** (const [char\\_type](#) &\_\_c1, const [char\\_type](#) &\_\_c2)
- static bool **eq** (const [char\\_type](#) &\_\_c1, const [char\\_type](#) &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const [char\\_type](#) \* **find** (const [char\\_type](#) \*\_\_s, std::size\_t \_\_n, const [char\\_type](#) &\_\_a)
- static const [char\\_type](#) \* **find** (const [char\\_type](#) \*\_\_s, size\_t \_\_n, const [char\\_type](#) &\_\_a)
- static constexpr const [char\\_type](#) \* **find** (const [char\\_type](#) \*\_\_s, std::size\_t \_\_n, const [char\\_type](#) &\_\_a)
- static constexpr std::size\_t **length** (const [char\\_type](#) \*\_\_s)
- static constexpr std::size\_t **length** (const [char\\_type](#) \*\_\_s)
- static size\_t **length** (const [char\\_type](#) \*\_\_s)
- static constexpr bool **lt** (const [char\\_type](#) &\_\_c1, const [char\\_type](#) &\_\_c2)
- static constexpr bool **lt** (const [char\\_type](#) &\_\_c1, const [char\\_type](#) &\_\_c2)
- static bool **lt** (const [char\\_type](#) &\_\_c1, const [char\\_type](#) &\_\_c2)
- static constexpr [char\\_type](#) \* **move** ([char\\_type](#) \*\_\_s1, const [char\\_type](#) \*\_\_s2, std::size\_t \_\_n)
- static [char\\_type](#) \* **move** ([char\\_type](#) \*\_\_s1, const [char\\_type](#) \*\_\_s2, size\_t \_\_n)
- static constexpr [char\\_type](#) \* **move** ([char\\_type](#) \*\_\_s1, const [char\\_type](#) \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr [char\\_type](#) **to\_char\_type** (const int\_type &\_\_c)
- static constexpr [char\\_type](#) **to\_char\_type** (const int\_type &\_\_c)
- static [char\\_type](#) **to\_char\_type** (const int\_type &\_\_i)
- static constexpr int\_type **to\_int\_type** (const [char\\_type](#) &\_\_c)
- static constexpr int\_type **to\_int\_type** (const [char\\_type](#) &\_\_c)
- static int\_type **to\_int\_type** (const [char\\_type](#) &\_\_c)

### 5.273.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St>
struct std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >
```

char\_traits<\_\_gnu\_cxx::character> specialization.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 5.274 std::char\_traits< wchar\_t > Struct Reference

```
#include <char_traits.h>
```

Inheritance diagram for std::char\_traits< wchar\_t >:



### Public Types

- typedef wchar\_t **char\_type**
- typedef wchar\_t **char\_type**
- using **comparison\_category**
- using **comparison\_category**
- typedef \_Char\_types< wchar\_t >::int\_type **int\_type**
- typedef wint\_t **int\_type**
- typedef \_Char\_types< wchar\_t >::off\_type **off\_type**
- typedef [streamoff](#) **off\_type**
- typedef \_Char\_types< wchar\_t >::pos\_type **pos\_type**
- typedef [wstreampos](#) **pos\_type**
- typedef \_Char\_types< wchar\_t >::state\_type **state\_type**
- typedef mbstate\_t **state\_type**

### Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr int\_type **eof** () noexcept
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept

- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2) noexcept
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c) noexcept
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c) noexcept
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c) noexcept

### 5.274.1 Detailed Description

#### 21.1.3.2 char\_traits specializations

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.275 \_\_gnu\_cxx::character<\_Value, \_Int, \_St > Struct Template Reference

```
#include <pod_char_traits.h>
```

### Public Types

- typedef [character](#)<\_Value, \_Int, \_St > **char\_type**
- typedef \_Int **int\_type**
- typedef \_St **state\_type**
- typedef \_Value **value\_type**

### Static Public Member Functions

- template<typename V2>  
static [char\\_type](#) **from** (const V2 &v)
- template<typename V2>  
static V2 **to** (const [char\\_type](#) &c)

### Public Attributes

- value\_type **value**

### 5.275.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St = std::mbstate_t>
struct __gnu_cxx::character< _Value, _Int, _St >
```

A POD class that serves as a character abstraction class.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 5.276 std::chi\_squared\_distribution<\_RealType> Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **chi\_squared\_distribution** (\_RealType \_\_n)
- **chi\_squared\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- \_RealType **n** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

### Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::chi\\_squared\\_distribution](#)< \_RealType1 > &\_\_x)
- bool **operator==** (const [chi\\_squared\\_distribution](#) &\_\_d1, const [chi\\_squared\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::chi\\_squared\\_distribution](#)< \_RealType1 > &\_\_x)



### 5.276.1 Detailed Description

```
template<typename _RealType = double>
class std::chi_squared_distribution< _RealType >
```

A chi\_squared\_distribution random number distribution.

The formula for the normal probability mass function is  $p(x|n) = \frac{x^{(n/2)-1}e^{-x/2}}{\Gamma(n/2)2^{n/2}}$

Since

C++11

### 5.276.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::chi_squared_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 5.276.3 Member Function Documentation

#### max()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::chi_squared_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

#### param() [1/2]

```
template<typename _RealType = double>
param_type std::chi_squared_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### param() [2/2]

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::chi_squared_distribution<_RealType>::reset () [inline]
Resets the distribution state.
```

**5.276.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::chi_squared_distribution<_RealType1> & __x) [friend]
Inserts a chi_squared_distribution random number distribution __x into the output stream __os.
```

**Parameters**

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                   |
| <code>__x</code>  | A <code>chi_squared_distribution</code> random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
    const chi_squared_distribution<_RealType> & __d1,
    const chi_squared_distribution<_RealType> & __d2) [friend]
```

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (
    std::basic_istream<_CharT, _Traits> & __is,
    std::chi_squared_distribution<_RealType1> & __x) [friend]
Extracts a chi_squared_distribution random number distribution __x from the input stream __is.
```

**Parameters**

|                   |                                                                         |
|-------------------|-------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                        |
| <code>__x</code>  | A <code>chi_squared_distribution</code> random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.277 std::codecvt<\_InternT,\_ExternT,\_StateT> Class Template Reference**

```
#include <codecvt.h>
```

Inheritance diagram for `std::codecvt<_InternT,_ExternT,_StateT>`:

**Public Types**

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

**Public Member Functions**

- **codecvt** (`_c_locale __cloc`, `size_t __refs=0`)
- **codecvt** (`size_t __refs=0`)

- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to↔\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

#### 5.277.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt<_InternT, _ExternT, _StateT >
```

Primary class template codecvt.

NB: Generic, mostly useless implementation.

#### 5.277.2 Member Function Documentation

##### do\_always\_noconv()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual bool std::codecvt<_InternT, _ExternT, _StateT >::do_always_noconv () const throw ( )
[protected], [virtual]
Implements std::\_\_codecvt\_abstract\_base<_InternT, _ExternT, _StateT >.
```

**do\_encoding()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< \_InternT, \_ExternT, \_StateT >::do\_encoding () const throw ( )    [protected],
[virtual]
Implements std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >.
```

**do\_in()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< \_InternT, \_ExternT, \_StateT >::do\_in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const    [protected], [virtual]
Implements std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >.
```

**do\_length()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< \_InternT, \_ExternT, \_StateT >::do\_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const    [protected], [virtual]
Implements std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >.
```

**do\_max\_length()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< \_InternT, \_ExternT, \_StateT >::do\_max\_length () const throw ( )    [protected],
[virtual]
Implements std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >.
```

**do\_out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< \_InternT, \_ExternT, \_StateT >::do\_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const    [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

**See also**

[do\\_out](#) for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, \\_StateT >](#).

**do\_unshift()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<_InternT, _ExternT, _StateT >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**in()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::out (
    state_type & __state,
```

```

const intern_type * __from,
const intern_type * __from_end,
const intern_type *& __from_next,
extern_type * __to,
extern_type * __to_end,
extern_type *& __to_next) const [inline], [inherited]

```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

References [do\\_out\(\)](#).

#### unshift()

```

template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]

```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

## Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.278 `std::codecvt<_InternT, _ExternT, encoding_state >` Class Template Reference

```
#include <codecvt_specializations.h>
```

Inheritance diagram for `std::codecvt<_InternT, _ExternT, encoding_state >`:



## Public Types

- `typedef state_type::descriptor_type` **descriptor\_type**
- `typedef _ExternT` **extern\_type**
- `typedef _ExternT` **extern\_type**
- `typedef _InternT` **intern\_type**
- `typedef _InternT` **intern\_type**
- `typedef codecvt_base::result` **result**
- `typedef codecvt_base::result` **result**
- `typedef` [encoding\\_state](#) **state\_type**
- `typedef` [\\_\\_gnu\\_cxx::encoding\\_state](#) **state\_type**



## Public Member Functions

- **codecvt** (`__c_locale __cloc`, `size_t __refs=0`)
- **codecvt** (`size_t __refs=0`)
- **codecvt** (`size_t __refs=0`)
- **codecvt** (`state_type &__enc`, `size_t __refs=0`)
- **bool always\_noconv** () const throw ()
- **bool always\_noconv** () const throw ()
- **int encoding** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state`, `const extern_type *__from`, `const extern_type *__from_end`, `const extern_type *__&__from_next`, `intern_type *__to`, `intern_type *__to_end`, `intern_type *__&__to_next`) const
- **result in** (`state_type &__state`, `const extern_type *__from`, `const extern_type *__from_end`, `const extern_type *__&__from_next`, `intern_type *__to`, `intern_type *__to_end`, `intern_type *__&__to_next`) const
- **int length** (`state_type &__state`, `const extern_type *__from`, `const extern_type *__end`, `size_t __max`) const
- **int length** (`state_type &__state`, `const extern_type *__from`, `const extern_type *__end`, `size_t __max`) const
- **int max\_length** () const throw ()
- **int max\_length** () const throw ()
- **result out** (`state_type &__state`, `const intern_type *__from`, `const intern_type *__from_end`, `const intern_type *__&__from_next`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *__&__to_next`) const
- **result out** (`state_type &__state`, `const intern_type *__from`, `const intern_type *__from_end`, `const intern_type *__&__from_next`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *__&__to_next`) const
- **result unshift** (`state_type &__state`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *__&__to_next`) const
- **result unshift** (`state_type &__state`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *__&__to_next`) const

## Static Public Attributes

- static `locale::id` **id**
- static `locale::id` **id**

## Protected Member Functions

- virtual **bool do\_always\_noconv** () const throw ()
- virtual **bool do\_always\_noconv** () const throw ()
- virtual **int do\_encoding** () const throw ()
- virtual **int do\_encoding** () const throw ()
- virtual **result do\_in** (`state_type &__state`, `const extern_type *__from`, `const extern_type *__from_end`, `const extern_type *__&__from_next`, `intern_type *__to`, `intern_type *__to_end`, `intern_type *__&__to_next`) const
- virtual **result do\_in** (`state_type &__state`, `const extern_type *__from`, `const extern_type *__from_end`, `const extern_type *__&__from_next`, `intern_type *__to`, `intern_type *__to_end`, `intern_type *__&__to_next`) const
- virtual **int do\_length** (`state_type &`, `const extern_type *__from`, `const extern_type *__end`, `size_t __max`) const
- virtual **int do\_length** (`state_type &`, `const extern_type *__from`, `const extern_type *__end`, `size_t __max`) const
- virtual **int do\_max\_length** () const throw ()
- virtual **int do\_max\_length** () const throw ()
- virtual **result do\_out** (`state_type &__state`, `const intern_type *__from`, `const intern_type *__from_end`, `const intern_type *__&__from_next`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *__&__to_next`) const
- virtual **result do\_out** (`state_type &__state`, `const intern_type *__from`, `const intern_type *__from_end`, `const intern_type *__&__from_next`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *__&__to_next`) const
- virtual **result do\_unshift** (`state_type &__state`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *__&__to←__next`) const
- virtual **result do\_unshift** (`state_type &__state`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *__&__to←__next`) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

#### 5.278.1 Detailed Description

template<typename \_InternT, typename \_ExternT>  
class std::codecvt<\_InternT, \_ExternT, encoding\_state >

codecvt<InternT, \_ExternT, encoding\_state> specialization.

#### 5.278.2 Member Function Documentation

##### do\_always\_noconv() [1/2]

```
virtual bool std::codecvt<_InternT, _ExternT, encoding_state >::do_always_noconv () const throw  
( ) [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

##### do\_always\_noconv() [2/2]

```
template<typename _InternT, typename _ExternT>  
bool std::codecvt<_InternT, _ExternT, encoding_state >::do_always_noconv () const throw ( )  
[protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

##### do\_encoding() [1/2]

```
virtual int std::codecvt<_InternT, _ExternT, encoding_state >::do_encoding () const throw ( )  
[protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

##### do\_encoding() [2/2]

```
template<typename _InternT, typename _ExternT>  
int std::codecvt<_InternT, _ExternT, encoding_state >::do_encoding () const throw ( ) [protected],  
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_in()** [1/2]

```
virtual result std::codecvt< _InternT, _ExternT, encoding_state >::do_in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type * __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type * __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_in()** [2/2]

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type * __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type * __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_length()** [1/2]

```
virtual int std::codecvt< _InternT, _ExternT, encoding_state >::do_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_length()** [2/2]

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_max\_length()** [1/2]

```
virtual int std::codecvt< _InternT, _ExternT, encoding_state >::do_max_length () const throw ( )
[protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_max\_length()** [2/2]

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_max_length () const throw ( ) [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_out()** [1/2]

```
virtual result std::codecvt<_InternT, _ExternT, encoding_state >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_out()** [2/2]

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt<_InternT, _ExternT, encoding_state >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_unshift()** [1/2]

```
virtual result std::codecvt<_InternT, _ExternT, encoding_state >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_unshift()** [2/2]

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt<_InternT, _ExternT, encoding_state >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

**in()** [1/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**in()** [2/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

#### `out()` [1/2]

```
result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**out() [2/2]**

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**unshift() [1/2]**

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

#### unshift() [2/2]

```
result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)



## 5.279 std::codecvt< char, char, mbstate\_t > Class Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt< char, char, mbstate\_t >:



### Public Types

- typedef char **extern\_type**
- typedef char **extern\_type**
- typedef char **intern\_type**
- typedef char **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const

- int **max\_length** () const throw ()
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

### Static Public Attributes

- static locale::id **id**
- static locale::id **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__c_locale __M_c_locale_codecvt`
- `__c_locale __M_c_locale_codecvt`

## Friends

- class `messages< char >`

### 5.279.1 Detailed Description

class `codecvt<char, char, mbstate_t>` specialization.

### 5.279.2 Member Function Documentation

#### `do_always_noconv()` [1/2]

virtual bool `std::codecvt< char, char, mbstate_t >::do_always_noconv () const throw ( )` [protected], [virtual]

Implements `std::__codecvt_abstract_base< char, char, mbstate_t >`.

#### `do_always_noconv()` [2/2]

virtual bool `std::codecvt< char, char, mbstate_t >::do_always_noconv () const throw ( )` [protected], [virtual]

Implements `std::__codecvt_abstract_base< char, char, mbstate_t >`.

#### `do_encoding()` [1/2]

virtual int `std::codecvt< char, char, mbstate_t >::do_encoding () const throw ( )` [protected], [virtual]

Implements `std::__codecvt_abstract_base< char, char, mbstate_t >`.

#### `do_encoding()` [2/2]

virtual int `std::codecvt< char, char, mbstate_t >::do_encoding () const throw ( )` [protected], [virtual]

Implements `std::__codecvt_abstract_base< char, char, mbstate_t >`.

#### `do_in()` [1/2]

virtual result `std::codecvt< char, char, mbstate_t >::do_in (`  
    `state_type & __state,`  
    `const extern_type * __from,`  
    `const extern_type * __from_end,`  
    `const extern_type *& __from_next,`  
    `intern_type * __to,`  
    `intern_type * __to_end,`  
    `intern_type *& __to_next) const` [protected], [virtual]

Implements `std::__codecvt_abstract_base< char, char, mbstate_t >`.

#### `do_in()` [2/2]

virtual result `std::codecvt< char, char, mbstate_t >::do_in (`  
    `state_type & __state,`  
    `const extern_type * __from,`

```

    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [protected], [virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

#### **do\_length()** [1/2]

```

virtual int std::codecvt< char, char, mbstate\_t >::do\_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

#### **do\_length()** [2/2]

```

virtual int std::codecvt< char, char, mbstate\_t >::do\_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

#### **do\_max\_length()** [1/2]

```

virtual int std::codecvt< char, char, mbstate\_t >::do\_max\_length () const throw () [protected],
[virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

#### **do\_max\_length()** [2/2]

```

virtual int std::codecvt< char, char, mbstate\_t >::do\_max\_length () const throw () [protected],
[virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

#### **do\_out()** [1/2]

```

virtual result std::codecvt< char, char, mbstate\_t >::do\_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]

```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

`out` for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

**do\_out()** [2/2]

```
virtual result std::codecvt< char, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

**See also**

[do\\_out\(\)](#) out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

**do\_unshift()** [1/2]

```
virtual result std::codecvt< char, char, mbstate_t >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

**do\_unshift()** [2/2]

```
virtual result std::codecvt< char, char, mbstate_t >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

**in()** [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do\_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

#### `in()` [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**out()** [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type * __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type * __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**out()** [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type * __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type * __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

#### `unshift()` [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore `state` to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |



**Returns**

codecvt\_base::result.

**unshift() [2/2]**

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

**5.280 std::codecvt< char16\_t, char, mbstate\_t > Class Reference**

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt< char16\_t, char, mbstate\_t >:



### Public Types

- typedef char **extern\_type**
- typedef char **extern\_type**
- typedef char16\_t **intern\_type**
- typedef char16\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- int **max\_length** () const throw ()

- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Public Attributes

- static **locale::id** id
- static **locale::id** id

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to←\_\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to←\_\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

### 5.280.1 Detailed Description

Class `codecvt<char16_t, char, mbstate_t>` specialization.  
 Converts between UTF-16 and UTF-8.

### 5.280.2 Member Function Documentation

#### `do_always_noconv()` [1/2]

```
virtual bool std::codecvt< char16\_t, char, mbstate\_t >::do\_always\_noconv \(\) const throw ( )    [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

#### `do_always_noconv()` [2/2]

```
virtual bool std::codecvt< char16\_t, char, mbstate\_t >::do\_always\_noconv \(\) const throw ( )    [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

#### `do_encoding()` [1/2]

```
virtual int std::codecvt< char16\_t, char, mbstate\_t >::do\_encoding \(\) const throw ( )    [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

#### `do_encoding()` [2/2]

```
virtual int std::codecvt< char16\_t, char, mbstate\_t >::do\_encoding \(\) const throw ( )    [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

#### `do_in()` [1/2]

```
virtual result std::codecvt< char16\_t, char, mbstate\_t >::do\_in \(
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const    [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

#### `do_in()` [2/2]

```
virtual result std::codecvt< char16\_t, char, mbstate\_t >::do\_in \(
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const    [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

**do\_length()** [1/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

**do\_length()** [2/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

**do\_max\_length()** [1/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_max_length () const throw ( ) [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

**do\_max\_length()** [2/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_max_length () const throw ( ) [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

**do\_out()** [1/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type * __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type * __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

[do\\_out\(\)](#) for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

**do\_out()** [2/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type * __from_next,
```

```
extern_type * __to,
extern_type * __to_end,
extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

[do\\_unshift\(\)](#) out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

### **do\_unshift()** [1/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

### **do\_unshift()** [2/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

### **in()** [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__state</code> | Persistent conversion state data. |
|----------------------|-----------------------------------|

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**in()** [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::in (  
    state_type & __state,  
    const extern_type * __from,  
    const extern_type * __from_end,  
    const extern_type *& __from_next,  
    intern_type * __to,  
    intern_type * __to_end,  
    intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**out()** [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**out()** [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.



The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

#### unshift() [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

**unshift()** [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

**5.281 std::codecvt< char32\_t, char, mbstate\_t > Class Reference**

```
#include <codecvt.h>
```

Inheritance diagram for `std::codecvt< char32_t, char, mbstate_t >`:



### Public Types

- typedef char **extern\_type**
- typedef char **extern\_type**
- typedef char32\_t **intern\_type**
- typedef char32\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- int **max\_length** () const throw ()

- result [out](#) (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result [out](#) (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result [unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result [unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Public Attributes

- static [locale::id](#) id
- static [locale::id](#) id

### Protected Member Functions

- virtual bool [do\\_always\\_noconv](#) () const throw ()
- virtual bool [do\\_always\\_noconv](#) () const throw ()
- virtual int [do\\_encoding](#) () const throw ()
- virtual int [do\\_encoding](#) () const throw ()
- virtual result [do\\_in](#) (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual result [do\\_in](#) (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual int [do\\_length](#) (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int [do\\_length](#) (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int [do\\_max\\_length](#) () const throw ()
- virtual int [do\\_max\\_length](#) () const throw ()
- virtual result [do\\_out](#) (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result [do\\_out](#) (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result [do\\_unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result [do\\_unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_codecvt](#)

### 5.281.1 Detailed Description

Class `codecvt<char32_t, char, mbstate_t>` specialization.  
Converts between UTF-32 and UTF-8.

### 5.281.2 Member Function Documentation

#### **do\_always\_noconv()** [1/2]

```
virtual bool std::codecvt< char32\_t, char, mbstate\_t >::do\_always\_noconv \(\) const throw \( \) [protected],  
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

#### **do\_always\_noconv()** [2/2]

```
virtual bool std::codecvt< char32\_t, char, mbstate\_t >::do\_always\_noconv \(\) const throw \( \) [protected],  
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

#### **do\_encoding()** [1/2]

```
virtual int std::codecvt< char32\_t, char, mbstate\_t >::do\_encoding \(\) const throw \( \) [protected],  
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

#### **do\_encoding()** [2/2]

```
virtual int std::codecvt< char32\_t, char, mbstate\_t >::do\_encoding \(\) const throw \( \) [protected],  
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

#### **do\_in()** [1/2]

```
virtual result std::codecvt< char32\_t, char, mbstate\_t >::do\_in \(  
    state_type & __state,  
    const extern_type * __from,  
    const extern_type * __from_end,  
    const extern_type *& __from_next,  
    intern_type * __to,  
    intern_type * __to_end,  
    intern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

#### **do\_in()** [2/2]

```
virtual result std::codecvt< char32\_t, char, mbstate\_t >::do\_in \(  
    state_type & __state,  
    const extern_type * __from,  
    const extern_type * __from_end,  
    const extern_type *& __from_next,  
    intern_type * __to,  
    intern_type * __to_end,  
    intern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

**do\_length()** [1/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**do\_length()** [2/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**do\_max\_length()** [1/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_max_length () const throw ( ) [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**do\_max\_length()** [2/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_max_length () const throw ( ) [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**do\_out()** [1/2]

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

`out` for more information.

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**do\_out()** [2/2]

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
```

```
extern_type * __to,
extern_type * __to_end,
extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

### do\_unshift() [1/2]

```
virtual result std::__codecvt< char32_t, char, mbstate_t >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

### do\_unshift() [2/2]

```
virtual result std::__codecvt< char32_t, char, mbstate_t >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

### in() [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do\_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__state</code> | Persistent conversion state data. |
|----------------------|-----------------------------------|

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

codecvt\_base::result.

## in() [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

codecvt\_base::result.



**out()** [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**out()** [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

#### `unshift()` [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

**unshift()** [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

**5.282 std::codecvt< wchar\_t, char, mbstate\_t > Class Reference**

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt< wchar\_t, char, mbstate\_t >:



### Public Types

- typedef char **extern\_type**
- typedef char **extern\_type**
- typedef wchar\_t **intern\_type**
- typedef wchar\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \* \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \* \_\_to\_next) const
- result **in** (state\_type &\_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \* \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \* \_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- int **length** (state\_type &\_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()

- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Public Attributes

- static **locale::id** id
- static **locale::id** id

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

**Protected Attributes**

- `__c_locale __M_c_locale_codecvt`
- `__c_locale __M_c_locale_codecvt`

**Friends**

- class `messages< wchar_t >`

**5.282.1 Detailed Description**

Class `codecvt<wchar_t, char, mbstate_t>` specialization.

Converts between narrow and wide characters in the native character set

**5.282.2 Member Function Documentation****do\_always\_noconv()** [1/2]

```
virtual bool std::codecvt< wchar_t, char, mbstate_t >::do_always_noconv () const throw ( )    [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

**do\_always\_noconv()** [2/2]

```
virtual bool std::codecvt< wchar_t, char, mbstate_t >::do_always_noconv () const throw ( )    [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

**do\_encoding()** [1/2]

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_encoding () const throw ( )    [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

**do\_encoding()** [2/2]

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_encoding () const throw ( )    [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

**do\_in()** [1/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const    [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

**do\_in()** [2/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_in (
    state_type & __state,
```

```

const extern_type * __from,
const extern_type * __from_end,
const extern_type *& __from_next,
intern_type * __to,
intern_type * __to_end,
intern_type *& __to_next) const [protected], [virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

#### **do\_length()** [1/2]

```

virtual int std::codecvt< wchar_t, char, mbstate_t >::do_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

#### **do\_length()** [2/2]

```

virtual int std::codecvt< wchar_t, char, mbstate_t >::do_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

#### **do\_max\_length()** [1/2]

```

virtual int std::codecvt< wchar_t, char, mbstate_t >::do_max_length () const throw ( ) [protected],
[virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

#### **do\_max\_length()** [2/2]

```

virtual int std::codecvt< wchar_t, char, mbstate_t >::do_max_length () const throw ( ) [protected],
[virtual]

```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

#### **do\_out()** [1/2]

```

virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]

```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

[do\\_out](#) for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

**do\_out()** [2/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

`do_in` for more information.

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

**do\_unshift()** [1/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

**do\_unshift()** [2/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

**in()** [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.



The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

#### `in()` [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

codecvt\_base::result.

## out() [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

codecvt\_base::result.

## out() [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

#### `unshift()` [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore `state` to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**unshift() [2/2]**

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

**5.283 std::codecvt\_base Class Reference**

```
#include <codecvt.h>
```

Inheritance diagram for `std::codecvt_base`:



## Public Types

- enum **result** { **ok** , **partial** , **error** , **noconv** }

### 5.283.1 Detailed Description

Empty base class for `codecvt` facet [22.2.1.5].

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.284 `std::codecvt_byname<_InternT, _ExternT, _StateT>` Class Template Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >:



### Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

### Public Member Functions

- **codecvt\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **codecvt\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `bool do_always_noconv ()` `const throw ()`
- virtual `int do_encoding ()` `const throw ()`
- virtual `result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type * &__from_next, intern_type *__to, intern_type *__to_end, intern_type * &__to_next)` `const`
- virtual `int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max)` `const`
- virtual `int do_max_length ()` `const throw ()`
- virtual `result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type * &__from_next, extern_type *__to, extern_type *__to_end, extern_type * &__to_next)` `const`
- virtual `result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type * &__to←__next)` `const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc)` `throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `const char * _S_get_c_name ()` `throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Protected Attributes

- `__c_locale _M_c_locale_codecvt`

### 5.284.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt_byname< _InternT, _ExternT, _StateT >
```

class `codecvt_byname` [22.2.1.6].

### 5.284.2 Member Function Documentation

#### `do_always_noconv()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual bool std::codecvt< _InternT, _ExternT, _StateT >::do_always_noconv () const throw ( )
[protected], [virtual], [inherited]
Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.
```

#### `do_encoding()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_encoding () const throw ( ) [protected],
[virtual], [inherited]
Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.
```

**do\_in()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<\_InternT, \_ExternT, \_StateT >::do\_in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type * __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type * __to_next) const [protected], [virtual], [inherited]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_length()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt<\_InternT, \_ExternT, \_StateT >::do\_length (
    state_type & ,
    const extern_type * __from,
    const extern_type * __end,
    size_t __max) const [protected], [virtual], [inherited]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_max\_length()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt<\_InternT, \_ExternT, \_StateT >::do\_max\_length () const throw ( ) [protected],
[virtual], [inherited]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<\_InternT, \_ExternT, \_StateT >::do\_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type * __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type * __to_next) const [protected], [virtual], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

`do_out` for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_unshift()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<\_InternT, \_ExternT, \_StateT >::do\_unshift (
    state_type & __state,
    extern_type * __to,
```



```
extern_type * __to_end,
extern_type *& __to_next) const [protected], [virtual], [inherited]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

### in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

### Returns

`codecvt_base::result`.

### out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

References [do\\_out\(\)](#).

#### unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__state</code>  | Persistent conversion state data. |
| <code>__to</code>     | Start of output buffer.           |
| <code>__to_end</code> | End of output buffer.             |

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__to_next</code> | Returns start of unused output area. |
|------------------------|--------------------------------------|

#### Returns

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

### 5.285 `std::__cxx11::collate<_CharT>` Class Template Reference

```
#include <locale_classes.h>
```

Inheritance diagram for `std::__cxx11::collate<_CharT>`:



#### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

#### Public Member Functions

- [collate](#) (`__c_locale __cloc`, `size_t __refs=0`)
- [collate](#) (`size_t __refs=0`)
- `int` [\\_M\\_compare](#) (`const _CharT *`, `const _CharT *`) `const throw ()`
- `size_t` [\\_M\\_transform](#) (`_CharT *`, `const _CharT *`, `size_t`) `const throw ()`
- `int` [compare](#) (`const _CharT *__lo1`, `const _CharT *__hi1`, `const _CharT *__lo2`, `const _CharT *__hi2`) `const`
- `long` [hash](#) (`const _CharT *__lo`, `const _CharT *__hi`) `const`
- [string\\_type transform](#) (`const _CharT *__lo`, `const _CharT *__hi`) `const`

#### Static Public Attributes

- static [locale::id](#) `id`

**Protected Member Functions**

- virtual [~collate](#) ()
- virtual int [do\\_compare](#) (const \_CharT \* \_\_lo1, const \_CharT \* \_\_hi1, const \_CharT \* \_\_lo2, const \_CharT \* \_\_hi2) const
- virtual long [do\\_hash](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const
- virtual [string\\_type do\\_transform](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const

**Static Protected Member Functions**

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_type\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

**Protected Attributes**

- \_\_c\_locale [\\_M\\_c\\_locale\\_collate](#)

**5.285.1 Detailed Description**

**template<typename \_CharT>**  
**class std::\_\_cxx11::collate<\_CharT>**

Facet for localized string comparison.

This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

**5.285.2 Member Typedef Documentation****char\_type**

```
template<typename _CharT>
typedef _CharT std::__cxx11::collate<_CharT>::char_type
```

Public typedefs.

**string\_type**

```
template<typename _CharT>
typedef basic_string<_CharT> std::__cxx11::collate<_CharT>::string_type
```

Public typedefs.

**5.285.3 Constructor & Destructor Documentation****collate()** [1/2]

```
template<typename _CharT>
std::__cxx11::collate<_CharT>::collate (
    size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

Referenced by [do\\_compare\(\)](#), [do\\_hash\(\)](#), and [do\\_transform\(\)](#).

**collate() [2/2]**

```
template<typename _CharT>
std::__cxx11::collate< _CharT >::collate (
    __c_locale __cloc,
    size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | The C locale.                   |
| <code>__refs</code> | Passed to the base facet class. |

References [std::locale::facet::facet\(\)](#).

**~collate()**

```
template<typename _CharT>
virtual std::__cxx11::collate< _CharT >::~~collate () [inline], [protected], [virtual]
Destructor.
```

**5.285.4 Member Function Documentation****compare()**

```
template<typename _CharT>
int std::__cxx11::collate< _CharT >::compare (
    const _CharT * __lo1,
    const _CharT * __hi1,
    const _CharT * __lo2,
    const _CharT * __hi2) const [inline]
```

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

**Returns**

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

References [do\\_compare\(\)](#).

**do\_compare()**

```
template<typename _CharT>
int std::collate<_CharT>::do_compare (
    const _CharT * __lo1,
    const _CharT * __hi1,
    const _CharT * __lo2,
    const _CharT * __hi2) const [protected], [virtual]
```

Compare two strings.

This function is a hook for derived classes to change the value returned.

See also

`compare()`.

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

**Returns**

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

References [collate\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::data\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::length\(\)](#).

Referenced by [compare\(\)](#).

**do\_hash()**

```
template<typename _CharT>
long std::collate<_CharT>::do_hash (
    const _CharT * __lo,
    const _CharT * __hi) const [protected], [virtual]
```

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

**Parameters**

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

**Returns**

Hash value.

References [collate\(\)](#).

Referenced by [hash\(\)](#).

**do\_transform()**

```
template<typename _CharT>
collate< _CharT >::string_type std::collate< _CharT >::do_transform (
    const _CharT * __lo,
    const _CharT * __hi) const [protected], [virtual]
```

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

**Parameters**

|                           |        |
|---------------------------|--------|
| $\leftrightarrow$<br>__lo | Start. |
| $\leftrightarrow$<br>__hi | End.   |

**Returns**

transformed string.

References [collate\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::append\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::c\\_str\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::clear\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::data\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::push\\_back\(\)](#).

Referenced by [transform\(\)](#).

**hash()**

```
template<typename _CharT>
long std::__cxx11::collate< _CharT >::hash (
    const _CharT * __lo,
    const _CharT * __hi) const [inline]
```

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning [collate::do\\_hash\(\)](#).

**Parameters**

|                           |                  |
|---------------------------|------------------|
| $\leftrightarrow$<br>__lo | Start of string. |
| $\leftrightarrow$<br>__hi | End of string.   |

**Returns**

Hash value.

References [do\\_hash\(\)](#).

**transform()**

```
template<typename _CharT>
string_type std::__cxx11::collate< _CharT >::transform (
    const _CharT * __lo,
    const _CharT * __hi) const [inline]
```

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

#### Parameters

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

#### Returns

Transformed `string_type`.

References [do\\_transform\(\)](#).

### 5.285.5 Member Data Documentation

#### id

```
template<typename _CharT>
locale::id std::__cxx11::collate<_CharT>::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 5.286 `std::__cxx11::collate_byname<_CharT>` Class Template Reference

```
#include <locale_classes.h>
```

### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

### Public Member Functions

- **collate\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **collate\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- **\_M\_compare** (const char \*, const char \*) const throw()
- **\_M\_compare** (const wchar\_t \*, const wchar\_t \*) const throw()
- **\_M\_transform** (char \*, const char \*, size\_t) const throw()
- **\_M\_transform** (wchar\_t \*, const wchar\_t \*, size\_t) const throw()

### 5.286.1 Detailed Description

```
template<typename _CharT>
class std::__cxx11::collate_byname<_CharT>
```

class `collate_byname` [22.2.4.2].



## 5.286.2 Member Typedef Documentation

### char\_type

```
template<typename _CharT>
typedef _CharT std::__cxx11::collate_byname< _CharT >::char_type
Public typedefs.
```

### string\_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::__cxx11::collate_byname< _CharT >::string_type
Public typedefs.
```

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 5.287 std::common\_iterator< \_It, \_Sent > Class Template Reference

```
#include <stl_iterator.h>
```

### Public Member Functions

- constexpr **common\_iterator** (*\_It* \_\_i) noexcept(is\_nothrow\_move\_constructible\_v< *\_It* >)
- constexpr **common\_iterator** (*\_Sent* \_\_s) noexcept(is\_nothrow\_move\_constructible\_v< *\_Sent* >)
- **common\_iterator** (**common\_iterator** &&)=default
- constexpr **common\_iterator** (**common\_iterator** &&\_\_x) noexcept(\_S\_noexcept< *\_It*, *\_Sent* >())
- **common\_iterator** (const **common\_iterator** &)=default
- constexpr **common\_iterator** (const **common\_iterator** &\_\_x) noexcept(\_S\_noexcept< const *\_It* &, const *\_Sent* & >())
- template<typename *\_It2*, typename *\_Sent2*>
 requires convertible\_to<const *\_It2*&, *\_It*> && convertible\_to<const *\_Sent2*&, *\_Sent*>
 constexpr **common\_iterator** (const **common\_iterator**< *\_It2*, *\_Sent2* > &\_\_x) noexcept(\_S\_noexcept< const *\_It2* &, const *\_Sent2* & >())
- constexpr decltype(auto) **operator\*** ()
- constexpr decltype(auto) **operator\*** () const
- constexpr **common\_iterator** & **operator++** ()
- constexpr decltype(auto) **operator++** (int)
- constexpr auto **operator->** () const
- constexpr **common\_iterator** & **operator=** (**common\_iterator** &&)=default
- constexpr **common\_iterator** & **operator=** (**common\_iterator** &&\_\_x) noexcept(is\_nothrow\_move\_assignable\_v< *\_It* > &&is\_nothrow\_move\_assignable\_v< *\_Sent* > &&is\_nothrow\_move\_constructible\_v< *\_It* > &&is\_nothrow\_move\_constructible\_v< *\_Sent* >)
- constexpr **common\_iterator** & **operator=** (const **common\_iterator** &)=default
- constexpr **common\_iterator** & **operator=** (const **common\_iterator** &\_\_x) noexcept(is\_nothrow\_copy\_assignable\_v< *\_It* > &&is\_nothrow\_copy\_assignable\_v< *\_Sent* > &&is\_nothrow\_copy\_constructible\_v< *\_It* > &&is\_nothrow\_copy\_constructible\_v< *\_Sent* >)
- template<typename *\_It2*, typename *\_Sent2*>
 requires convertible\_to<const *\_It2*&, *\_It*> && convertible\_to<const *\_Sent2*&, *\_Sent*> && assignable\_from<*\_It*&, const *\_It2*&> && assignable\_from<*\_Sent*&, const *\_Sent2*&>
 constexpr **common\_iterator** & **operator=** (const **common\_iterator**< *\_It2*, *\_Sent2* > &\_\_x) noexcept(is\_nothrow\_constructible\_v< *\_It*, const *\_It2* & > &&is\_nothrow\_constructible\_v< *\_Sent*, const *\_Sent2* & > &&is\_nothrow\_assignable\_v< *\_It* &, const *\_It2* & > &&is\_nothrow\_assignable\_v< *\_Sent* &, const *\_Sent2* & >)

**Friends**

- constexpr iter\_rvalue\_reference\_t< \_It > **iter\_move** (const common\_iterator &\_\_i) noexcept(noexcept(ranges::iter\_move(std::declval< const \_It & >()))))
- template<indirectly\_swappable< \_It > \_It2, typename \_Sent2>  
constexpr void **iter\_swap** (const common\_iterator &\_\_x, const common\_iterator< \_It2, \_Sent2 > &\_\_y) noexcept(noexcept(ranges::iter\_swap(std::declval< const \_It & >(), std::declval< const \_It2 & >()))))
- template<sized\_sentinel\_for< \_It > \_It2, sized\_sentinel\_for< \_It > \_Sent2>  
requires sized\_sentinel\_for< \_Sent, \_It2>  
constexpr iter\_difference\_t< \_It2 > **operator-** (const common\_iterator &\_\_x, const common\_iterator< \_It2, \_Sent2 > &\_\_y)
- template<typename \_It2, sentinel\_for< \_It > \_Sent2>  
requires sentinel\_for< \_Sent, \_It2>  
constexpr bool **operator==** (const common\_iterator &\_\_x, const common\_iterator< \_It2, \_Sent2 > &\_\_y)
- template<typename \_It2, sentinel\_for< \_It > \_Sent2>  
requires sentinel\_for< \_Sent, \_It2> && equality\_comparable\_with< \_It, \_It2>  
constexpr bool **operator==** (const common\_iterator &\_\_x, const common\_iterator< \_It2, \_Sent2 > &\_\_y)

**5.287.1 Detailed Description**

```
template<input_or_output_iterator _It, sentinel_for< _It > _Sent>
requires (Isame_as< _It, _Sent>) && copyable< _It>
class std::common_iterator< _It, _Sent >
```

An iterator/sentinel adaptor for representing a non-common range.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

**5.288 std::common\_type< \_Tp > Struct Template Reference****5.288.1 Detailed Description**

```
template<typename... _Tp>
struct std::common_type< _Tp >
```

common\_type

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.289 std::common\_type< chrono::duration< \_Rep, \_Period > > Struct Template Reference**

```
#include <chrono.h>
```

**Public Types**

- using **type**

**5.289.1 Detailed Description**

```
template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period > >
```

Specialization of common\_type for one chrono::duration type.

The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 5.290 `std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >` Struct Template Reference

```
#include <chrono.h>
```

#### Public Types

- using `type`

#### 5.290.1 Detailed Description

```
template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >
```

Specialization of `common_type` for two identical `chrono::duration` types.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 5.291 `std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >` Struct Template Reference

```
#include <chrono.h>
```

#### 5.291.1 Detailed Description

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
struct std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >
```

Specialization of `common_type` for `chrono::duration` types.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 5.292 `std::common_type< chrono::time_point< _Clock, _Duration > >` Struct Template Reference

```
#include <chrono.h>
```

#### Public Types

- using `type`

#### 5.292.1 Detailed Description

```
template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration > >
```

Specialization of `common_type` for one `chrono::time_point` type.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 5.293 `std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >` Struct Template Reference

```
#include <chrono.h>
```

## Public Types

- using `type`

### 5.293.1 Detailed Description

```
template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration
> >
```

Specialization of `common_type` for two identical `chrono::time_point` types.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

## 5.294 `std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >` Struct Template Reference

```
#include <chrono.h>
```

### 5.294.1 Detailed Description

```
template<typename _Clock, typename _Duration1, typename _Duration2>
struct std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >
```

Specialization of `common_type` for `chrono::time_point` types.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

## 5.295 `std::compare_three_way_result< _Tp, _Up >` Struct Template Reference

```
#include <compare>
```

### 5.295.1 Detailed Description

```
template<typename _Tp, typename _Up = _Tp>
struct std::compare_three_way_result< _Tp, _Up >
```

[`cmp.result`], result of three-way comparison  
The documentation for this struct was generated from the following file:

- [compare](#)

## 5.296 `std::complex< _Tp >` Class Template Reference

```
#include <complex>
```

## Public Types

- typedef `_Tp` [value\\_type](#)

## Public Member Functions

- constexpr `complex` (const `_Tp` &\_\_r=\_Tp(), const `_Tp` &\_\_i=\_Tp())
- constexpr `complex` (const `complex` &)=default
- template<typename `_Up`>  
constexpr `complex` (const `complex`< `_Up` > &\_\_z)
- `__attribute` ((\_\_abi\_tag\_\_("cxx11"))) const expr `_Tp` imag() const
- `__attribute` ((\_\_abi\_tag\_\_("cxx11"))) const expr `_Tp` real() const
- constexpr `complex` `__rep` () const
- constexpr void `imag` (`_Tp` \_\_val)
- constexpr `complex`< `_Tp` > & `operator*=` (const `_Tp` &)
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator*=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator+=` (const `_Tp` &\_\_t)
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator+=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator-=` (const `_Tp` &\_\_t)
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator-=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator/=` (const `_Tp` &)
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator/=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator=` (const `_Tp` &)
- constexpr `complex` & `operator=` (const `complex` &)=default
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator=` (const `complex`< `_Up` > &)
- constexpr void `real` (`_Tp` \_\_val)

### 5.296.1 Detailed Description

template<typename `_Tp`>  
class std::complex< `_Tp` >

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

#### Parameters

|                 |                                    |
|-----------------|------------------------------------|
| <code>Tp</code> | Type of real and imaginary values. |
|-----------------|------------------------------------|

### 5.296.2 Member Typedef Documentation

#### value\_type

template<typename `_Tp`>  
typedef `_Tp` `std::complex`< `_Tp` >::value\_type  
Value typedef.

### 5.296.3 Constructor & Destructor Documentation

#### `complex()` [1/2]

```
template<typename _Tp>
std::complex< _Tp >::complex (
    const _Tp & __r = _Tp(),
    const _Tp & __i = _Tp()) [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

#### `complex()` [2/2]

```
template<typename _Tp>
template<typename _Up>
std::complex< _Tp >::complex (
    const complex< _Up > & __z) [inline], [constexpr]
```

Converting constructor.

### 5.296.4 Member Function Documentation

#### `operator+=()`

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator+= (
    const _Tp & __t) [inline], [constexpr]
```

Add a scalar to this complex number.

#### `operator-=()`

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator-= (
    const _Tp & __t) [inline], [constexpr]
```

Subtract a scalar from this complex number.

The documentation for this class was generated from the following file:

- [complex](#)

## 5.297 `std::complex< double >` Class Reference

```
#include <complex>
```

### Public Types

- typedef `__complex__ double` `_ComplexT`
- typedef double `value_type`
- typedef double `value_type`

### Public Member Functions

- constexpr `complex` (`_ComplexT __z`)
- constexpr `complex` (const `complex` &)=default
- constexpr `complex` (const `complex` &)=default
- constexpr `complex` (const `complex`< `_Up` > &\_\_z)
- constexpr `complex` (const `complex`< float > &\_\_z)
- constexpr `complex` (const `complex`< long double > &)
- constexpr `complex` (const double &\_\_r=double(), const double &\_\_i=double())

- constexpr **complex** (double \_\_r=0.0, double \_\_i=0.0)
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr double imag() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr double imag() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr double real() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr double real() const
- constexpr **complex** **\_\_rep** () const
- constexpr **\_ComplexT** **\_\_rep** () const
- constexpr void **imag** (double \_\_val)
- constexpr void **imag** (double \_\_val)
- template<typename \_Tp>  
constexpr **complex** & **operator\*=** (const **complex**<\_Tp> &\_\_z)
- constexpr **complex**<double> & **operator\*=** (const **complex**<\_Up> &)
- constexpr **complex**<double> & **operator\*=** (const double &)
- constexpr **complex** & **operator\*=** (double \_\_d)
- template<typename \_Tp>  
constexpr **complex** & **operator+=** (const **complex**<\_Tp> &\_\_z)
- constexpr **complex**<double> & **operator+=** (const **complex**<\_Up> &)
- constexpr **complex**<double> & **operator+=** (const double &\_\_t)
- constexpr **complex** & **operator+=** (double \_\_d)
- template<typename \_Tp>  
constexpr **complex** & **operator-=** (const **complex**<\_Tp> &\_\_z)
- constexpr **complex**<double> & **operator-=** (const **complex**<\_Up> &)
- constexpr **complex**<double> & **operator-=** (const double &\_\_t)
- constexpr **complex** & **operator-=** (double \_\_d)
- template<typename \_Tp>  
constexpr **complex** & **operator/=** (const **complex**<\_Tp> &\_\_z)
- constexpr **complex**<double> & **operator/=** (const **complex**<\_Up> &)
- constexpr **complex**<double> & **operator/=** (const double &)
- constexpr **complex** & **operator/=** (double \_\_d)
- constexpr **complex** & **operator=** (const **complex** &)=default
- constexpr **complex** & **operator=** (const **complex** &)=default
- template<typename \_Tp>  
constexpr **complex** & **operator=** (const **complex**<\_Tp> &\_\_z)
- constexpr **complex**<double> & **operator=** (const **complex**<\_Up> &)
- constexpr **complex**<double> & **operator=** (const double &)
- constexpr **complex** & **operator=** (double \_\_d)
- constexpr void **real** (double \_\_val)
- constexpr void **real** (double \_\_val)

### 5.297.1 Detailed Description

26.2.3 complex specializations **complex**<double> specialization

### 5.297.2 Member Typedef Documentation

#### value\_type

```
typedef double std::complex<double>::value_type
```

Value typedef.

### 5.297.3 Constructor & Destructor Documentation

#### complex() [1/2]

```
std::complex< double >::complex (
    const double & __r = _Tp(),
    const double & __i = _Tp()) [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

#### complex() [2/2]

```
std::complex< double >::complex (
    const complex< _Up > & __z) [inline], [constexpr]
```

Converting constructor.

### 5.297.4 Member Function Documentation

#### operator\*=( ) [1/2]

```
complex< double > & std::complex< double >::operator*= (
    const complex< _Up > & __z) [constexpr]
```

Multiply this complex number by another.

#### operator\*=( ) [2/2]

```
complex< double > & std::complex< double >::operator*= (
    const double & __t) [constexpr]
```

Multiply this complex number by a scalar.

#### operator+=( ) [1/2]

```
complex< double > & std::complex< double >::operator+= (
    const complex< _Up > & __z) [constexpr]
```

Add another complex number to this one.

#### operator+=( ) [2/2]

```
complex< double > & std::complex< double >::operator+= (
    const double & __t) [inline], [constexpr]
```

Add a scalar to this complex number.

#### operator-=( ) [1/2]

```
complex< double > & std::complex< double >::operator-= (
    const complex< _Up > & __z) [constexpr]
```

Subtract another complex number from this one.

#### operator-=( ) [2/2]

```
complex< double > & std::complex< double >::operator-= (
    const double & __t) [inline], [constexpr]
```

Subtract a scalar from this complex number.



**operator/=( ) [1/2]**

```
complex< double > & std::complex< double >::operator/= (
    const complex< _Up > & __z) [constexpr]
```

Divide this complex number by another.

**operator/=( ) [2/2]**

```
complex< double > & std::complex< double >::operator/= (
    const double & __t) [constexpr]
```

Divide this complex number by a scalar.

**operator=( ) [1/2]**

```
complex< double > & std::complex< double >::operator= (
    const complex< _Up > & __z) [constexpr]
```

Assign another complex number to this one.

**operator=( ) [2/2]**

```
complex< double > & std::complex< double >::operator= (
    const double & __t) [constexpr]
```

Assign a scalar to this complex number.

The documentation for this class was generated from the following file:

- [complex](#)

**5.298 std::complex< float > Class Reference**

```
#include <complex>
```

**Public Types**

- typedef \_\_complex\_\_ float **\_ComplexT**
- typedef float [value\\_type](#)
- typedef float **value\_type**

**Public Member Functions**

- constexpr **complex** ( \_ComplexT \_\_z)
- constexpr **complex** (const [complex](#) &)=default
- constexpr **complex** (const [complex](#) &)=default
- constexpr [complex](#) (const [complex](#)< \_Up > &\_\_z)
- constexpr **complex** (const [complex](#)< double > &)
- constexpr **complex** (const [complex](#)< long double > &)
- constexpr [complex](#) (const float &\_\_r=float(), const float &\_\_i=float())
- constexpr **complex** (float \_\_r=0.0f, float \_\_i=0.0f)
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr float imag() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr float imag() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr float real() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr float real() const
- constexpr [complex](#) **\_\_rep** () const
- constexpr \_ComplexT **\_\_rep** () const
- constexpr void **imag** (float \_\_val)

- constexpr void **imag** (float \_\_val)
- template<class \_Tp>  
constexpr `complex` & **operator\*=** (const `complex`< \_Tp > &\_\_z)
- constexpr `complex`< float > & **operator\*=** (const `complex`< \_Up > &)
- constexpr `complex`< float > & **operator\*=** (const float &)
- constexpr `complex` & **operator\*=** (float \_\_f)
- template<typename \_Tp>  
constexpr `complex` & **operator+=** (const `complex`< \_Tp > &\_\_z)
- constexpr `complex`< float > & **operator+=** (const `complex`< \_Up > &)
- constexpr `complex`< float > & **operator+=** (const float &\_\_t)
- constexpr `complex` & **operator+=** (float \_\_f)
- template<class \_Tp>  
constexpr `complex` & **operator-=** (const `complex`< \_Tp > &\_\_z)
- constexpr `complex`< float > & **operator-=** (const `complex`< \_Up > &)
- constexpr `complex`< float > & **operator-=** (const float &\_\_t)
- constexpr `complex` & **operator-=** (float \_\_f)
- template<class \_Tp>  
constexpr `complex` & **operator/=** (const `complex`< \_Tp > &\_\_z)
- constexpr `complex`< float > & **operator/=** (const `complex`< \_Up > &)
- constexpr `complex`< float > & **operator/=** (const float &)
- constexpr `complex` & **operator/=** (float \_\_f)
- constexpr `complex` & **operator=** (const `complex` &)=default
- constexpr `complex` & **operator=** (const `complex` &)=default
- template<typename \_Tp>  
constexpr `complex` & **operator=** (const `complex`< \_Tp > &\_\_z)
- constexpr `complex`< float > & **operator=** (const `complex`< \_Up > &)
- constexpr `complex`< float > & **operator=** (const float &)
- constexpr `complex` & **operator=** (float \_\_f)
- constexpr void **real** (float \_\_val)
- constexpr void **real** (float \_\_val)

### 5.298.1 Detailed Description

26.2.3 complex specializations `complex<float>` specialization

### 5.298.2 Member Typedef Documentation

#### value\_type

```
typedef float std::complex< float >::value_type
```

Value typedef.

### 5.298.3 Constructor & Destructor Documentation

#### complex() [1/2]

```
std::complex< float >::complex (
    const float & __r = _Tp(),
    const float & __i = _Tp()) [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

**complex()** [2/2]

```
std::complex< float >::complex (
    const complex< _Up > & __z) [inline], [constexpr]
```

Converting constructor.

**5.298.4 Member Function Documentation****operator\*=( )** [1/2]

```
complex< float > & std::complex< float >::operator*= (
    const complex< _Up > & __z) [constexpr]
```

Multiply this complex number by another.

**operator\*=( )** [2/2]

```
complex< float > & std::complex< float >::operator*= (
    const float & __t) [constexpr]
```

Multiply this complex number by a scalar.

**operator+=( )** [1/2]

```
complex< float > & std::complex< float >::operator+= (
    const complex< _Up > & __z) [constexpr]
```

Add another complex number to this one.

**operator+=( )** [2/2]

```
complex< float > & std::complex< float >::operator+= (
    const float & __t) [inline], [constexpr]
```

Add a scalar to this complex number.

**operator-=( )** [1/2]

```
complex< float > & std::complex< float >::operator-= (
    const complex< _Up > & __z) [constexpr]
```

Subtract another complex number from this one.

**operator-=( )** [2/2]

```
complex< float > & std::complex< float >::operator-= (
    const float & __t) [inline], [constexpr]
```

Subtract a scalar from this complex number.

**operator/=( )** [1/2]

```
complex< float > & std::complex< float >::operator/= (
    const complex< _Up > & __z) [constexpr]
```

Divide this complex number by another.

**operator/=( )** [2/2]

```
complex< float > & std::complex< float >::operator/= (
    const float & __t) [constexpr]
```

Divide this complex number by a scalar.

**operator=()** [1/2]

```
complex< float > & std::complex< float >::operator= (
    const complex< _Up > & __z) [constexpr]
```

Assign another complex number to this one.

**operator=()** [2/2]

```
complex< float > & std::complex< float >::operator= (
    const float & __t) [constexpr]
```

Assign a scalar to this complex number.

The documentation for this class was generated from the following file:

- [complex](#)

**5.299 std::complex< long double > Class Reference**

```
#include <complex>
```

**Public Types**

- typedef \_\_complex\_\_ long double **\_ComplexT**
- typedef long double [value\\_type](#)
- typedef long double **value\_type**

**Public Member Functions**

- constexpr **complex** (\_ComplexT \_\_z)
- constexpr **complex** (const [complex](#) &)=default
- constexpr **complex** (const [complex](#) &)=default
- constexpr [complex](#) (const [complex](#)< \_Up > &\_\_z)
- constexpr **complex** (const [complex](#)< double > &\_\_z)
- constexpr **complex** (const [complex](#)< float > &\_\_z)
- constexpr [complex](#) (const long double &\_\_r=long double(), const long double &\_\_i=long double())
- constexpr **complex** (long double \_\_r=0.0L, long double \_\_i=0.0L)
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr long double imag() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr long double imag() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr long double real() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr long double real() const
- constexpr [complex](#) **\_\_rep** () const
- constexpr \_ComplexT **\_\_rep** () const
- constexpr void **imag** (long double \_\_val)
- constexpr void **imag** (long double \_\_val)
- template<typename \_Tp>
 constexpr [complex](#) & **operator\*=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#)< long double > & **operator\*=** (const [complex](#)< \_Up > &)
- constexpr [complex](#)< long double > & **operator\*=** (const long double &)
- constexpr [complex](#) & **operator\*=** (long double \_\_r)
- template<typename \_Tp>
 constexpr [complex](#) & **operator+=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#)< long double > & **operator+=** (const [complex](#)< \_Up > &)
- constexpr [complex](#)< long double > & **operator+=** (const long double &\_\_t)
- constexpr [complex](#) & **operator+=** (long double \_\_r)

- `template<typename _Tp>`  
`constexpr complex & operator-= (const complex< _Tp > &__z)`
- `constexpr complex< long double > & operator-= (const complex< _Up > &)`
- `constexpr complex< long double > & operator-= (const long double &__t)`
- `constexpr complex & operator-= (long double __r)`
- `template<typename _Tp>`  
`constexpr complex & operator/= (const complex< _Tp > &__z)`
- `constexpr complex< long double > & operator/= (const complex< _Up > &)`
- `constexpr complex< long double > & operator/= (const long double &)`
- `constexpr complex & operator/= (long double __r)`
- `constexpr complex & operator= (const complex &)=default`
- `constexpr complex & operator= (const complex &)=default`
- `template<typename _Tp>`  
`constexpr complex & operator= (const complex< _Tp > &__z)`
- `constexpr complex< long double > & operator= (const complex< _Up > &)`
- `constexpr complex< long double > & operator= (const long double &)`
- `constexpr complex & operator= (long double __r)`
- `constexpr void real (long double __val)`
- `constexpr void real (long double __val)`

### 5.299.1 Detailed Description

26.2.3 complex specializations `complex<long double>` specialization

### 5.299.2 Member Typedef Documentation

#### value\_type

`typedef long double std::complex< long double >::value_type`  
 Value typedef.

### 5.299.3 Constructor & Destructor Documentation

#### `complex()` [1/2]

```
std::complex< long double >::complex (
    const long double & __r = _Tp(),
    const long double & __i = _Tp()) [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

#### `complex()` [2/2]

```
std::complex< long double >::complex (
    const complex< _Up > & __z) [inline], [constexpr]
```

Converting constructor.

### 5.299.4 Member Function Documentation

#### `operator*=( )` [1/2]

```
complex< long double > & std::complex< long double >::operator*= (
    const complex< _Up > & __z) [constexpr]
```

Multiply this complex number by another.

**operator\*=( )** [2/2]

```
complex< long double > & std::complex< long double >::operator*= (
    const long double & __t) [constexpr]
```

Multiply this complex number by a scalar.

**operator+=( )** [1/2]

```
complex< long double > & std::complex< long double >::operator+= (
    const complex< _Up > & __z) [constexpr]
```

Add another complex number to this one.

**operator+=( )** [2/2]

```
complex< long double > & std::complex< long double >::operator+= (
    const long double & __t) [inline], [constexpr]
```

Add a scalar to this complex number.

**operator-=( )** [1/2]

```
complex< long double > & std::complex< long double >::operator-= (
    const complex< _Up > & __z) [constexpr]
```

Subtract another complex number from this one.

**operator-=( )** [2/2]

```
complex< long double > & std::complex< long double >::operator-= (
    const long double & __t) [inline], [constexpr]
```

Subtract a scalar from this complex number.

**operator/=( )** [1/2]

```
complex< long double > & std::complex< long double >::operator/= (
    const complex< _Up > & __z) [constexpr]
```

Divide this complex number by another.

**operator/=( )** [2/2]

```
complex< long double > & std::complex< long double >::operator/= (
    const long double & __t) [constexpr]
```

Divide this complex number by a scalar.

**operator=( )** [1/2]

```
complex< long double > & std::complex< long double >::operator= (
    const complex< _Up > & __z) [constexpr]
```

Assign another complex number to this one.

**operator=( )** [2/2]

```
complex< long double > & std::complex< long double >::operator= (
    const long double & __t) [constexpr]
```

Assign a scalar to this complex number.

The documentation for this class was generated from the following file:

- [complex](#)

### 5.300 `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >` Class Template Reference

```
#include <cond_key_dtor_entry_dealtor.hpp>
```

#### Public Types

- typedef HT\_Map::entry **entry**
- typedef HT\_Map::entry\_allocator **entry\_allocator**
- typedef alloc\_traits::allocator\_type **entry\_allocator**
- typedef alloc\_traits::pointer **entry\_pointer**
- typedef HT\_Map::key\_type **key\_type**

#### Public Member Functions

- **cond\_dealtor** (entry\_allocator \*p\_a, entry \*p\_e)
- **cond\_dealtor** (entry\_pointer p\_e)
- void **set\_key\_destruct** ()
- void **set\_no\_action** ()
- void **set\_no\_action\_destructor** ()

#### Protected Attributes

- bool **m\_key\_destruct**
- entry\_allocator \*const **m\_p\_a**
- entry \*const **m\_p\_e**

#### 5.300.1 Detailed Description

```
template<typename Entry, typename _Alloc>  
class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >
```

Conditional deallocate constructor argument.

Conditional key destructor, cc\_hash.

The documentation for this class was generated from the following files:

- [cond\\_dealtor.hpp](#)
- [cond\\_key\\_dtor\\_entry\\_dealtor.hpp](#)

### 5.301 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >` Class Template Reference

```
#include <ov_tree_map_.hpp>
```

#### Public Member Functions

- **cond\_dtor** (value\_vector a\_vec, iterator &r\_last\_it, Size\_Type total\_size)
- void **set\_no\_action** ()

#### Protected Attributes

- value\_vector **m\_a\_vec**
- const Size\_Type **m\_max\_size**
- bool **m\_no\_action**
- iterator & **m\_r\_last\_it**

### 5.301.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
template<typename Size_Type>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor<
Size_Type >
```

Conditional destructor.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map.hpp](#)

## 5.302 \_\_gnu\_cxx::condition\_base Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::condition\_base:



### Public Member Functions

- **condition\_base** (const [condition\\_base](#) &)=default
- **condition\_base** & **operator=** (const [condition\\_base](#) &)=default

### 5.302.1 Detailed Description

Base struct for condition policy.

Requires a public member function with the signature void throw\_conditionally()

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.303 std::condition\_variable Class Reference

```
#include <condition_variable>
```

### Public Types

- typedef \_\_gthread\_cond\_t \* **native\_handle\_type**

### Public Member Functions

- **condition\_variable** (const [condition\\_variable](#) &)=delete
- native\_handle\_type **native\_handle** ()
- void **notify\_all** () noexcept



- void **notify\_one** () noexcept
- [condition\\_variable](#) & **operator=** (const [condition\\_variable](#) &)=delete
- void **wait** ([unique\\_lock](#)< [mutex](#) > &\_\_lock)
- template<typename \_Predicate>  
void **wait** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, \_Predicate \_\_p)
- template<typename \_Rep, typename \_Period>  
[cv\\_status](#) **wait\_for** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Rep, typename \_Period, typename \_Predicate>  
bool **wait\_for** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- template<typename \_Clock, typename \_Duration>  
[cv\\_status](#) **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Clock, typename \_Duration, typename \_Predicate>  
bool **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)
- template<typename \_Duration>  
[cv\\_status](#) **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< [steady\\_clock](#), \_Duration > &\_\_atime)
- template<typename \_Duration>  
[cv\\_status](#) **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< [system\\_clock](#), \_Duration > &\_\_atime)

### 5.303.1 Detailed Description

[condition\\_variable](#)

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 5.304 std::condition\_variable\_any Class Reference

```
#include <condition_variable>
```

### Public Member Functions

- **condition\_variable\_any** (const [condition\\_variable\\_any](#) &)=delete
- void **notify\_all** () noexcept
- void **notify\_one** () noexcept
- [condition\\_variable\\_any](#) & **operator=** (const [condition\\_variable\\_any](#) &)=delete
- template<typename \_Lock>  
void **wait** (\_Lock &\_\_lock)
- template<typename \_Lock, typename \_Predicate>  
void **wait** (\_Lock &\_\_lock, \_Predicate \_\_p)
- template<typename \_Lock, typename \_Rep, typename \_Period>  
[cv\\_status](#) **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Lock, typename \_Rep, typename \_Period, typename \_Predicate>  
bool **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- template<typename \_Lock, typename \_Clock, typename \_Duration>  
[cv\\_status](#) **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Lock, typename \_Clock, typename \_Duration, typename \_Predicate>  
bool **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)

### 5.304.1 Detailed Description

`condition_variable_any`

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 5.305 `std::conditional<_Cond, _Iftrue, _Iffalse >` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using `type`

### 5.305.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse>
struct std::conditional<_Cond, _Iftrue, _Iffalse >
```

Define a member typedef `type` to one of two argument types.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.306 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata >::const_iterator` Struct Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata >::const_iterator`:



### Public Types

- typedef `_Alloc::difference_type` **difference\_type**

- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef node\_pointer\_pointer **pointer**
- typedef node\_pointer\_reference **reference**
- typedef node\_pointer **value\_type**

#### Public Member Functions

- **const\_iterator** (node\_pointer\_pointer p\_p\_cur=0, node\_pointer\_pointer p\_p\_end=0)
- bool **operator!=** (const [const\\_iterator](#) &other) const
- node\_const\_pointer **operator\*** () const
- [const\\_iterator](#) & **operator++** ()
- [const\\_iterator](#) **operator++** (int)
- const node\_pointer\_pointer **operator->** () const
- bool **operator==** (const [const\\_iterator](#) &other) const

#### Public Attributes

- node\_pointer\_pointer **m\_p\_p\_cur**
- node\_pointer\_pointer **m\_p\_p\_end**

#### 5.306.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata >::const_iterator
```

Constant child iterator.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

#### 5.307 std::chrono::tzdb\_list::const\_iterator Class Reference

```
#include <chrono>
```

#### Public Types

- using **difference\_type**
- using **iterator\_category**
- using **pointer**
- using **reference**
- using **value\_type**

#### Public Member Functions

- **const\_iterator** (const [const\\_iterator](#) &)=default
- **const\_iterator** ([const\\_iterator](#) &&)=default
- reference **operator\*** () const noexcept
- [const\\_iterator](#) & **operator++** ()
- [const\\_iterator](#) **operator++** (int)
- pointer **operator->** () const noexcept
- [const\\_iterator](#) & **operator=** (const [const\\_iterator](#) &)=default
- [const\\_iterator](#) & **operator=** ([const\\_iterator](#) &&)=default
- bool **operator==** (const [const\\_iterator](#) &) const noexcept=default

## Friends

- class `tzdb_list`

### 5.307.1 Detailed Description

An iterator into the `tzdb_list`

As a extension, in libstdc++ each `tzdb` is reference-counted and the `const_iterator` type shares ownership of the object it refers to. This ensures that a `tzdb` erased from the list will not be destroyed while there is an iterator that refers to it.

The documentation for this class was generated from the following file:

- [chrono](#)

## 5.308 std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg > Class Template Reference

```
#include <std_function.h>
```

Inheritance diagram for `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

## Public Member Functions

- `const_mem_fun1_ref_t` (`_Ret` (`_Tp::*` `__pf`) (`_Arg`) `const`)
- `_Ret operator()` (`const _Tp &` `__r`, `_Arg` `__x`) `const`

### 5.308.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

### 5.308.2 Member Typedef Documentation

#### first\_argument\_type

typedef `_Tp` `std::binary_function`< `_Tp`, `_Arg`, `_Ret` >::`first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

#### result\_type

typedef `_Ret` `std::binary_function`< `_Tp`, `_Arg`, `_Ret` >::`result_type` [inherited]

`result_type` is the return type

#### second\_argument\_type

typedef `_Arg` `std::binary_function`< `_Tp`, `_Arg`, `_Ret` >::`second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 5.309 std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`:



#### Public Types

- typedef `const _Tp *` `first_argument_type`
- typedef `_Ret` `result_type`
- typedef `_Arg` `second_argument_type`

#### Public Member Functions

- `const_mem_fun1_t` (`_Ret` (`_Tp::*`\_\_pf) (`_Arg`) const)
- `_Ret operator()` (const `_Tp *`\_\_p, `_Arg` \_\_x) const

### 5.309.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

### 5.309.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef const _Tp * std::binary_function< const _Tp *, _Arg, _Ret >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
typedef _Ret std::binary_function< const _Tp *, _Arg, _Ret >::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
typedef _Arg std::binary_function< const _Tp *, _Arg, _Ret >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.310 std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >:



### Public Types

- typedef \_Tp [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

## Public Member Functions

- `const_mem_fun_ref_t` (`_Ret(_Tp::*__pf)() const`)
- `_Ret operator()` (`const _Tp &__r`) const

### 5.310.1 Detailed Description

```
template<typename _Ret, typename _Tp>
class std::const_mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

### 5.310.2 Member Typedef Documentation

#### argument\_type

```
typedef _Tp std::unary_function< _Tp, _Ret >::argument_type [inherited]
argument_type is the type of the argument
```

#### result\_type

```
typedef _Ret std::unary_function< _Tp, _Ret >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.311 std::const\_mem\_fun\_t< \_Ret, \_Tp > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::const_mem_fun_t< _Ret, _Tp >`:



## Public Types

- `typedef const _Tp *` [argument\\_type](#)
- `typedef _Ret` [result\\_type](#)

**Public Member Functions**

- `const_mem_fun_t` (`_Ret(_Tp::*__pf)()` const)
- `_Ret operator()` (const `_Tp *` `__p`) const

**5.311.1 Detailed Description**

```
template<typename _Ret, typename _Tp>
class std::const_mem_fun_t<_Ret, _Tp>
```

One of the [adaptors for member pointers](#).

**5.311.2 Member Typedef Documentation****argument\_type**

```
typedef const _Tp * std::unary_function< const _Tp *, _Ret >::argument_type [inherited]
argument_type is the type of the argument
```

**result\_type**

```
typedef _Ret std::unary_function< const _Tp *, _Ret >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**5.312 `__gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2>` Struct Template Reference**

```
#include <functional>
```

**Public Types**

- typedef `_Arg1` **first\_argument\_type**
- typedef `_Result` **result\_type**
- typedef `_Arg2` **second\_argument\_type**

**Public Member Functions**

- `constant_binary_fun` (const `_Result &` `__v`)
- const `result_type &` **operator()** (const `_Arg1 &`, const `_Arg2 &`) const

**Public Attributes**

- `_Result` **\_M\_val**

**5.312.1 Detailed Description**

```
template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1>
struct __gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2>
```

An [SGI extension](#) .

The documentation for this struct was generated from the following file:

- [ext/functional](#)



### 5.313 `__gnu_parallel::constant_size_blocks_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::constant_size_blocks_tag`:



#### 5.313.1 Detailed Description

Selects the constant block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.314 `__gnu_cxx::constant_unary_fun<_Result, _Argument>` Struct Template Reference

```
#include <functional>
```

#### Public Types

- typedef `_Argument` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- **constant\_unary\_fun** (`const _Result &__v`)
- `const result_type &` **operator()** (`const _Argument &`) `const`

#### Public Attributes

- `result_type` **\_M\_val**

#### 5.314.1 Detailed Description

```
template<class _Result, class _Argument = _Result>
struct __gnu_cxx::constant_unary_fun<_Result, _Argument>
```

An [SGI extension](#).

The documentation for this struct was generated from the following file:

- [ext/functional](#)

### 5.315 `__gnu_cxx::constant_void_fun<_Result>` Struct Template Reference

```
#include <functional>
```

#### Public Types

- typedef `_Result` **result\_type**

#### Public Member Functions

- **constant\_void\_fun** (const `_Result` &\_\_v)
- const `result_type` & **operator()** () const

#### Public Attributes

- `result_type` **\_M\_val**

#### 5.315.1 Detailed Description

```
template<class _Result>
struct __gnu_cxx::constant_void_fun<_Result>
```

An [SGI extension](#) .

The documentation for this struct was generated from the following file:

- [ext/functional](#)

### 5.316 `__gnu_pbds::detail::container_base_dispatch<Key, Mapped, _Alloc, Tag, Policy_Tl>` Struct Template Reference

#### 5.316.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_Tl = null_type>
struct __gnu_pbds::detail::container_base_dispatch<Key, Mapped, _Alloc, Tag, Policy_Tl>
```

Dispatch mechanism, primary template for associative types.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.317 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type>` Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

#### Public Types

- typedef [binary\\_heap](#)<\_VTp, Cmp\_Fn, \_Alloc> [type](#)

#### 5.317.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type>
```

Specialization for `binary_heap`.

### 5.317.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef binary\_heap<_VTp, Cmp_Fn, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch< _VTp,
Cmp_Fn, _Alloc, binary\_heap\_tag, null\_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 5.318 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< \_VTp, Cmp\_Fn, \_Alloc, [binomial\\_heap\\_tag](#), [null\\_type](#) > Struct Template Reference

```
#include <priority\_queue\_base\_dispatch.hpp>
```

#### Public Types

- typedef [binomial\\_heap](#)< \_VTp, Cmp\_Fn, \_Alloc > [type](#)

#### 5.318.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< _VTp, Cmp_Fn, _Alloc, binomial\_heap\_tag, null\_type >
```

Specialization for [binomial\\_heap](#).

### 5.318.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef binomial\_heap<_VTp, Cmp_Fn, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch< _VTp,
Cmp_Fn, _Alloc, binomial\_heap\_tag, null\_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 5.319 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< \_VTp, Cmp\_Fn, \_Alloc, [pairing\\_heap\\_tag](#), [null\\_type](#) > Struct Template Reference

```
#include <priority\_queue\_base\_dispatch.hpp>
```

#### Public Types

- typedef [pairing\\_heap](#)< \_VTp, Cmp\_Fn, \_Alloc > [type](#)

#### 5.319.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< _VTp, Cmp_Fn, _Alloc, pairing\_heap\_tag, null\_type >
```

Specialization for [pairing\\_heap](#).

### 5.319.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef pairing\_heap<_VTp, Cmp_Fn, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch< _VTp,
Cmp_Fn, _Alloc, pairing\_heap\_tag, null\_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 5.320 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

#### Public Types

- typedef [rc\\_binomial\\_heap](#)< \_VTp, Cmp\_Fn, \_Alloc > [type](#)

#### 5.320.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type
>
```

Specialization for rc\_binary\_heap.

### 5.320.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef rc\_binomial\_heap<_VTp, Cmp_Fn, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch< _VTp,
Cmp_Fn, _Alloc, rc\_binomial\_heap\_tag, null\_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 5.321 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

#### Public Types

- typedef [thin\\_heap](#)< \_VTp, Cmp\_Fn, \_Alloc > [type](#)

#### 5.321.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >
```

Specialization for thin\_heap.

### 5.321.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef thin\_heap<_VTp, Cmp_Fn, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch< _VTp, Cmp_Fn, _Alloc, thin\_heap\_tag, null\_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 5.322 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< Key, Mapped, \_Alloc, cc\_hash\_tag, Policy\_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [cc\\_ht\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc, at3t::value, at4t, at2t > [type](#)

#### 5.322.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >
```

Specialization colision-chaining hash map.

#### 5.322.2 Member Typedef Documentation

#### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef cc\_ht\_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, Mapped, _Alloc, cc\_hash\_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.323 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [gp\\_ht\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc, at3t::value, at4t, at5t, at2t > [type](#)

#### 5.323.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash map.

### 5.323.2 Member Typedef Documentation

#### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>
```

```
typedef gp\_ht\_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> \_\_gnu\_pbds::detail::container\_base\_dispatch< Key,
```

```
Key, Mapped, _Alloc, gp\_hash\_tag, Policy_TI >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.324 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [lu\\_map](#)< Key, Mapped, at0t, \_Alloc, at1t > [type](#)

#### 5.324.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>
```

```
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI >
```

Specialization for list-update map.

### 5.324.2 Member Typedef Documentation

#### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>
```

```
typedef lu\_map<Key, Mapped, at0t, _Alloc, at1t> \_\_gnu\_pbds::detail::container\_base\_dispatch< Key,
```

```
Mapped, _Alloc, list\_update\_tag, Policy_TI >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.325 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [ov\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > [type](#)

#### 5.325.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>
```

```
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI >
```

Specialization ordered-vector tree map.

### 5.325.2 Member Typedef Documentation

#### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef ov\_tree\_map<Key, Mapped, at0t, at1t, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, Mapped, _Alloc, ov\_tree\_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.326 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< Key, Mapped, \_Alloc, [pat\\_trie\\_tag](#), Policy\_Tl > Struct Template Reference

```
#include <container\_base\_dispatch.hpp>
```

#### Public Types

- typedef [pat\\_trie\\_map](#)< Key, Mapped, at1t, \_Alloc > **type**

#### 5.326.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, _Alloc, pat\_trie\_tag, Policy_TI >
```

Specialization for PATRICIA trie map.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.327 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< Key, Mapped, \_Alloc, [rb\\_tree\\_tag](#), Policy\_Tl > Struct Template Reference

```
#include <container\_base\_dispatch.hpp>
```

#### Public Types

- typedef [rb\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > **type**

#### 5.327.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, _Alloc, rb\_tree\_tag, Policy_TI >
```

Specialization for R-B tree map.

### 5.327.2 Member Typedef Documentation

#### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef rb\_tree\_map<Key, Mapped, at0t, at1t, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, Mapped, _Alloc, rb\_tree\_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.328 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

### Public Types

- typedef `splay_tree_map< Key, Mapped, at0t, at1t, _Alloc >` `type`

#### 5.328.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>  
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree map.

#### 5.328.2 Member Typedef Documentation

##### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>  
typedef splay_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<  
Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- `container_base_dispatch.hpp`

## 5.329 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

### Public Types

- typedef `cc_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t >` `type`

#### 5.329.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>  
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >
```

Specialization collision-chaining hash set.

#### 5.329.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>  
typedef cc_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_  
Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- `container_base_dispatch.hpp`



### 5.330 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `gp_ht_set< Key, null\_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t >` [type](#)

#### 5.330.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash set.

#### 5.330.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef gp_ht_set<Key, null\_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, null\_type, _Alloc, gp\_hash\_tag, Policy_Tl >::type
Dispatched type.
```

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.331 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `lu_set< Key, null\_type, at0t, _Alloc, at1t >` [type](#)

#### 5.331.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >
```

Specialization for list-update set.

#### 5.331.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef lu_set<Key, null\_type, at0t, _Alloc, at1t> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, null\_type, _Alloc, list\_update\_tag, Policy_Tl >::type
Dispatched type.
```

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.332 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `ov_tree_set< Key, null_type, at0t, at1t, _Alloc >` `type`

#### 5.332.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >
```

Specialization ordered-vector tree set.

#### 5.332.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef ov_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.333 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `pat_trie_set< Key, null_type, at1t, _Alloc >` `type`

#### 5.333.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >
```

Specialization for PATRICIA trie set.

#### 5.333.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef pat_trie_set<Key, null_type, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.334 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `rb_tree_set< Key, null\_type, at0t, at1t, _Alloc >` `type`

#### 5.334.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >
```

Specialization for R-B tree set.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.335 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `splay_tree_set< Key, null\_type, at0t, at1t, _Alloc >` `type`

#### 5.335.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree set.

#### 5.335.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef splay_tree_set<Key, null\_type, at0t, at1t, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, null\_type, _Alloc, splay\_tree\_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.336 `__gnu_pbds::container_error` Struct Reference

```
#include <exception.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::container\_error:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 5.336.1 Detailed Description

Base class for exceptions.

#### 5.336.2 Member Function Documentation

##### what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

### 5.337 \_\_gnu\_pbds::container\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```



## Public Types

- enum { [order\\_preserving](#) , [erase\\_can\\_throw](#) , [split\\_join\\_can\\_throw](#) , [reverse\\_iteration](#) }
- typedef [container\\_traits\\_base](#)< container\_category > **base\_type**
- typedef Cntnr::container\_category **container\_category**
- typedef Cntnr **container\_type**
- typedef base\_type::invalidation\_guarantee **invalidation\_guarantee**

### 5.338.1 Detailed Description

```
template<typename Cntnr>
struct __gnu_pbds::container_traits< Cntnr >
```

Container traits.

### 5.338.2 Member Enumeration Documentation

#### anonymous enum

```
template<typename Cntnr>
anonymous enum
```

#### Enumerator

|                                   |                                                             |
|-----------------------------------|-------------------------------------------------------------|
| <code>order_preserving</code>     | True only if Cntnr objects guarantee storing keys by order. |
| <code>erase_can_throw</code>      | True only if erasing a key can throw.                       |
| <code>split_join_can_throw</code> | True only if split or join operations can throw.            |
| <code>reverse_iteration</code>    | True only reverse iterators are supported.                  |

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.339 `__gnu_pbds::container_traits_base<_Tag>` Struct Template Reference

### 5.339.1 Detailed Description

```
template<typename _Tag>
struct __gnu_pbds::container_traits_base<_Tag>
```

Primary template, container traits base.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.340 `__gnu_pbds::container_traits_base<binary_heap_tag>` Struct Reference

```
#include <tag_and_trait.hpp>
```

## Public Types

- enum { [order\\_preserving](#) , [erase\\_can\\_throw](#) , [split\\_join\\_can\\_throw](#) , [reverse\\_iteration](#) }
- typedef [binary\\_heap\\_tag](#) **container\_category**
- typedef [basic\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

### 5.340.1 Detailed Description

Specialization, binary heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.341 `__gnu_pbds::container_traits_base< binomial_heap_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef [binomial\\_heap\\_tag](#) `container_category`
- typedef [point\\_invalidation\\_guarantee](#) `invalidation_guarantee`

### 5.341.1 Detailed Description

Specialization, binomial heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.342 `__gnu_pbds::container_traits_base< cc_hash_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef [cc\\_hash\\_tag](#) `container_category`
- typedef [point\\_invalidation\\_guarantee](#) `invalidation_guarantee`

### 5.342.1 Detailed Description

Specialization, cc hash.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.343 `__gnu_pbds::container_traits_base< gp_hash_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef [gp\\_hash\\_tag](#) `container_category`
- typedef [basic\\_invalidation\\_guarantee](#) `invalidation_guarantee`

### 5.343.1 Detailed Description

Specialization, gp hash.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.344 `__gnu_pbds::container_traits_base< list_update_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

#### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef `list_update_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

#### 5.344.1 Detailed Description

Specialization, list update.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.345 `__gnu_pbds::container_traits_base< ov_tree_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

#### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef `ov_tree_tag` `container_category`
- typedef `basic_invalidation_guarantee` `invalidation_guarantee`

#### 5.345.1 Detailed Description

Specialization, ov tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.346 `__gnu_pbds::container_traits_base< pairing_heap_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

#### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef `pairing_heap_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

#### 5.346.1 Detailed Description

Specialization, pairing heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.347 `__gnu_pbds::container_traits_base< pat_trie_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```



### Public Types

- enum { **order\_preserving** , **erase\_can\_throw** , **split\_join\_can\_throw** , **reverse\_iteration** }
- typedef [pat\\_trie\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 5.347.1 Detailed Description

Specialization, pat trie.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.348 `__gnu_pbds::container_traits_base< rb_tree_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { **order\_preserving** , **erase\_can\_throw** , **split\_join\_can\_throw** , **reverse\_iteration** }
- typedef [rb\\_tree\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 5.348.1 Detailed Description

Specialization, rb tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.349 `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { **order\_preserving** , **erase\_can\_throw** , **split\_join\_can\_throw** , **reverse\_iteration** }
- typedef [rc\\_binomial\\_heap\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 5.349.1 Detailed Description

Specialization, rc binomial heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.350 `__gnu_pbds::container_traits_base< splay_tree_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { **order\_preserving** , **erase\_can\_throw** , **split\_join\_can\_throw** , **reverse\_iteration** }
- typedef [splay\\_tree\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 5.350.1 Detailed Description

Specialization, splay tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.351 `__gnu_pbds::container_traits_base< thin_heap_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

#### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef [thin\\_heap\\_tag](#) `container_category`
- typedef [point\\_invalidation\\_guarantee](#) `invalidation_guarantee`

#### 5.351.1 Detailed Description

Specialization, thin heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.352 `std::contiguous_iterator_tag` Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for `std::contiguous_iterator_tag`:



#### 5.352.1 Detailed Description

Contiguous iterators point to objects stored contiguously in memory.  
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 5.353 `std::copyable_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>` Class Template Reference

```
#include <functional>
```

##### Public Types

- using `result_type`

##### Public Member Functions

- [copyable\\_function](#) () noexcept

- `template<typename _Fn, typename _Vt = decay_t<_Fn>>`  
requires `(is_same_v<_Vt, copyable_function>) && (!is_in_place_type_v<_Vt>) && __is_callable_from<_Vt>`  
`copyable_function` (`_Fn &&__f`) `noexcept(_S_nothrow_init<_Vt, _Fn>())`
- `copyable_function` (`copyable_function &&__x`) `noexcept`
- `copyable_function` (`copyable_function const &__x`)
- `template<typename _Tp, typename... _Args>`  
requires `is_constructible_v<_Tp, _Args...> && __is_callable_from<_Tp>`  
`copyable_function` (`in_place_type_t<_Tp>, _Args &&... __args`) `noexcept(_S_nothrow_init<_Tp, _Args...>())`
- `template<typename _Tp, typename _Up, typename... _Args>`  
requires `is_constructible_v<_Tp, initializer_list<_Up>&, _Args...> && __is_callable_from<_Tp>`  
`copyable_function` (`in_place_type_t<_Tp>, initializer_list<_Up> __il, _Args &&... __args`) `noexcept(_S_nothrow_init<_Tp, initializer_list<_Up> &, _Args...>())`
- `copyable_function` (`nullptr_t`) `noexcept`
- `operator bool` () `const noexcept`
- `_Res operator()` (`_ArgTypes... __args`) `_GLIBCXX_MOF_CV noexcept(_Noex)`
- `template<typename _Fn>`  
requires `is_constructible_v<copyable_function, _Fn>`  
`copyable_function & operator=` (`_Fn &&__f`) `noexcept(is_nothrow_constructible_v<copyable_function, _Fn>)`
- `copyable_function & operator=` (`const copyable_function &__x`)
- `copyable_function & operator=` (`copyable_function &&__x`) `noexcept`
- `copyable_function & operator=` (`nullptr_t`) `noexcept`
- `void swap` (`copyable_function &__x`) `noexcept`

## Friends

- `template<typename _Func>`  
`auto & __polyfunc::__base_of` (`_Func &`) `noexcept`
- `template<typename _Func>`  
`auto & __polyfunc::__invoker_of` (`_Func &`) `noexcept`
- `template<typename _Dst, typename _Src>`  
`constexpr bool __polyfunc::__is_invoker_convertible` () `noexcept`
- `bool operator==` (`const copyable_function &__x, nullptr_t`) `noexcept`
- `void swap` (`copyable_function &__x, copyable_function &__y`) `noexcept`

### 5.353.1 Detailed Description

`template<typename _Res, typename... _ArgTypes, bool _Noex>`  
**class** `std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`

Polymorphic copyable function wrapper.

Since

C++26

The `std::copyable_function` class template is a call wrapper similar to `std::function`, but it does not provide information about its target, and preserves constness.

It also supports const-qualification, ref-qualification, and no-throw guarantees. The qualifications and exception-specification of the `copyable_function::operator()` member function are respected when invoking the target function.

## 5.353.2 Constructor & Destructor Documentation

### copyable\_function() [1/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function
() [inline], [noexcept]
```

Creates an empty object.

References [copyable\\_function\(\)](#).

Referenced by [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator==](#), [swap\(\)](#), and [swap](#).

### copyable\_function() [2/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
    nullptr_t ) [inline], [noexcept]
```

Creates an empty object.

References [copyable\\_function\(\)](#).

### copyable\_function() [3/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
    copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Moves the target object, leaving the source empty.

References [copyable\\_function\(\)](#).

### copyable\_function() [4/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
    copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> const & __x)
[inline]
```

Copies the target object.

References [copyable\\_function\(\)](#).

### copyable\_function() [5/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = decay_t<_Fn>>
requires (!is_same_v<_Vt, copyable_function>) && (!__is_in_place_type_v<_Vt>) && __is_callable<
    _from<_Vt>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
    _Fn && __f) [inline], [noexcept]
```

Stores a target object initialized from the argument.

References [copyable\\_function\(\)](#), and [std::forward\(\)](#).

### copyable\_function() [6/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename... _Args>
requires is_constructible_v<_Tp, _Args...> && __is_callable_from<_Tp>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
```

```
in_place_type_t< _Tp > ,
_Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.

References [copyable\\_function\(\)](#), and [std::forward\(\)](#).

### `copyable_function()` [7/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename _Up, typename... _Args>
requires is_constructible_v<_Tp, initializer_list<_Up>&, _Args...> && __is_callable_from<_Tp>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
    in_place_type_t< _Tp > ,
    initializer_list< _Up > __il,
    _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.

References [copyable\\_function\(\)](#), and [std::forward\(\)](#).

## 5.353.3 Member Function Documentation

### `operator bool()`

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator bool ()
const [inline], [explicit], [noexcept]
```

True if a target object is present, false otherwise.

### `operator()()`

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
_Res std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
    _ArgTypes... __args) [inline], [noexcept]
```

Invoke the target object.

The target object will be invoked using the supplied arguments, and as an lvalue or rvalue, and as const or non-const, as dictated by the template arguments of the `copyable_function` specialization.

#### Precondition

Must not be empty.

References [std::forward\(\)](#), and [operator\(\)\(\)](#).

Referenced by [operator\(\)\(\)](#).

### `operator=()` [1/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn>
requires is_constructible_v<copyable_function, _Fn>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
    _Fn && __f) [inline], [noexcept]
```

Stores a new target object, initialized from the argument.

References [copyable\\_function\(\)](#), [std::forward\(\)](#), and [operator=\(\)](#).

### `operator=()` [2/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
    _Res && __r) [inline], [noexcept]
```

```
Noex)>::operator= (
    const copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x)
[inline]
```

Stores a copy of the source target object.

References [copyable\\_function\(\)](#), and [operator=\(\)](#).

#### **operator=()** [3/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
    copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Stores a new target object, leaving x empty.

References [copyable\\_function\(\)](#), [std::addressof\(\)](#), and [operator=\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

#### **operator=()** [4/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
    nullptr_t ) [inline], [noexcept]
```

Destroys the target object (if any).

References [copyable\\_function\(\)](#), and [operator=\(\)](#).

#### **swap()**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::swap (
    copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x) [inline],
[noexcept]
```

Exchange the target objects (if any).

References [copyable\\_function\(\)](#), and [swap\(\)](#).

Referenced by [swap\(\)](#), and [swap](#).

### **5.353.4 Friends And Related Symbol Documentation**

#### **operator==**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
bool operator== (
    const copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
    nullptr_t ) [friend]
```

Check for emptiness by comparing with nullptr.

References [copyable\\_function\(\)](#), and [operator==](#).

Referenced by [operator==](#).

#### **swap**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void swap (
    copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
    copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __y) [friend]
```

Exchange the target objects (if any).

References [copyable\\_function\(\)](#), and [swap\(\)](#).

The documentation for this class was generated from the following file:

- [cpyfunc\\_impl.h](#)

## 5.354 std::counted\_iterator<\_It> Class Template Reference

```
#include <stl_iterator.h>
```

### Public Types

- using **difference\_type**
- using **iterator\_type**

### Public Member Functions

- constexpr **counted\_iterator** ([\\_It](#) \_\_i, [iter\\_difference\\_t](#)< [\\_It](#) > \_\_n)
- template<typename [\\_It2](#)>  
requires convertible\_to<const [\\_It2](#)&, [\\_It](#)>  
constexpr **counted\_iterator** (const [counted\\_iterator](#)< [\\_It2](#) > &\_\_x)
- constexpr [\\_It](#) **base** () &&noexcept(is\_nothrow\_move\_constructible\_v< [\\_It](#) >)
- constexpr const [\\_It](#) & **base** () const &noexcept
- constexpr [iter\\_difference\\_t](#)< [\\_It](#) > **count** () const noexcept
- constexpr decltype(auto) **operator\*** () const noexcept(noexcept(\*\_M\_current))
- constexpr decltype(auto) **operator\*** () noexcept(noexcept(\*\_M\_current))
- constexpr [counted\\_iterator](#) **operator+** ([iter\\_difference\\_t](#)< [\\_It](#) > \_\_n) const
- constexpr [counted\\_iterator](#) & **operator++** ()
- constexpr decltype(auto) **operator++** (int)
- constexpr [counted\\_iterator](#) **operator++** (int)
- constexpr [counted\\_iterator](#) & **operator+=** ([iter\\_difference\\_t](#)< [\\_It](#) > \_\_n)
- constexpr [counted\\_iterator](#) **operator-** ([iter\\_difference\\_t](#)< [\\_It](#) > \_\_n) const
- constexpr [counted\\_iterator](#) & **operator--** ()
- constexpr [counted\\_iterator](#) **operator--** (int)
- constexpr [counted\\_iterator](#) & **operator-=** ([iter\\_difference\\_t](#)< [\\_It](#) > \_\_n)
- constexpr auto **operator->** () const noexcept
- template<typename [\\_It2](#)>  
requires assignable\_from< [\\_It](#)&, const [\\_It2](#)&>  
constexpr [counted\\_iterator](#) & **operator=** (const [counted\\_iterator](#)< [\\_It2](#) > &\_\_x)
- constexpr decltype(auto) **operator[]** ([iter\\_difference\\_t](#)< [\\_It](#) > \_\_n) const noexcept(noexcept(\_M\_current[\_\_n]))

### Friends

- constexpr [iter\\_rvalue\\_reference\\_t](#)< [\\_It](#) > **iter\_move** (const [counted\\_iterator](#) &\_\_i) noexcept(noexcept(ranges::iter\_move(\_\_i.\_M\_current)))
- template<indirectly\_swappable< [\\_It](#) > [\\_It2](#)>  
constexpr void **iter\_swap** (const [counted\\_iterator](#) &\_\_x, const [counted\\_iterator](#)< [\\_It2](#) > &\_\_y) noexcept(noexcept(ranges::iter\_swap(\_\_x.\_M\_current, \_\_y.\_M\_current)))
- constexpr [counted\\_iterator](#) **operator+** ([iter\\_difference\\_t](#)< [\\_It](#) > \_\_n, const [counted\\_iterator](#) &\_\_x)
- template<common\_with< [\\_It](#) > [\\_It2](#)>  
constexpr [iter\\_difference\\_t](#)< [\\_It2](#) > **operator-** (const [counted\\_iterator](#) &\_\_x, const [counted\\_iterator](#)< [\\_It2](#) > &\_\_y) noexcept
- constexpr [iter\\_difference\\_t](#)< [\\_It](#) > **operator-** (const [counted\\_iterator](#) &\_\_x, [default\\_sentinel\\_t](#)) noexcept
- constexpr [iter\\_difference\\_t](#)< [\\_It](#) > **operator-** ([default\\_sentinel\\_t](#), const [counted\\_iterator](#) &\_\_y) noexcept



- `template<common_with<_It> _It2>`  
`constexpr strong_ordering operator<=> (const counted_iterator &__x, const counted_iterator<_It2> &__y)`  
`noexcept`
- `template<common_with<_It> _It2>`  
`constexpr bool operator== (const counted_iterator &__x, const counted_iterator<_It2> &__y) noexcept`
- `constexpr bool operator== (const counted_iterator &__x, default_sentinel_t) noexcept`

### 5.354.1 Detailed Description

`template<input_or_output_iterator _It>`  
**class** `std::counted_iterator<_It>`

An iterator adaptor that keeps track of the distance to the end.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 5.355 std::ctype<\_CharT> Class Template Reference

`#include <locale_facets.h>`

Inheritance diagram for `std::ctype<_CharT>`:



### Public Types

- `typedef const int * __to_type`
- `typedef _CharT char_type`
- `typedef __ctype_abstract_base<_CharT>::mask mask`

## Public Member Functions

- **ctype** (size\_t \_\_refs=0)
- const char\_type \* **is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- bool **is** (mask \_\_m, char\_type \_\_c) const
- char **narrow** (char\_type \_\_c, char \_\_default) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char\_type \* **scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* **scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static **locale::id** id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual const char\_type \* **do\_is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- virtual bool **do\_is** (mask \_\_m, char\_type \_\_c) const
- virtual char **do\_narrow** (char\_type, char \_\_default) const
- virtual const char\_type \* **do\_narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- virtual const char\_type \* **do\_scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* **do\_scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* **do\_tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_tolower** (char\_type \_\_c) const
- virtual const char\_type \* **do\_toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_toupper** (char\_type \_\_c) const
- virtual char\_type **do\_widen** (char \_\_c) const
- virtual const char \* **do\_widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_dest) const

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### 5.355.1 Detailed Description

**template<typename \_CharT>**  
**class std::ctype< \_CharT >**

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.

This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in `__ctype_abstract_base`, to allow for implementation flexibility. See `ctype<wchar_t>` for an example. The functions are documented in `__ctype_abstract_base`.

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

### 5.355.2 Member Function Documentation

**do\_is()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

Implements `std::__ctype_abstract_base< _CharT >`.

**do\_is()** [2/2]

```
template<typename _CharT>
virtual bool std::ctype< _CharT >::do_is (
```

```
mask __m,
char_type __c) const [protected], [virtual]
```

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                          |                                    |
|--------------------------|------------------------------------|
| $\leftrightarrow$<br>__c | The char_type to find the mask of. |
| $\leftrightarrow$<br>__m | The mask to compare against.       |

#### Returns

(M & \_\_m) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### do\_narrow() [1/2]

```
template<typename _CharT>
virtual char std::ctype<_CharT>::do_narrow (
    char_type __c,
    char __default) const [protected], [virtual]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, default is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|           |                                     |
|-----------|-------------------------------------|
| __c       | The char_type to convert.           |
| __default | Char to return if conversion fails. |

#### Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::narrow\(\)](#), and [std::ctype<char>::narrow\(\)](#).

#### do\_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [protected], [virtual]
```

Narrow char\_type array to char.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### do\_scan\_is()

```
template<typename _CharT>
virtual const char_type * std::__ctype<_CharT>::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Find `char_type` matching mask.

This function searches for and returns the first `char_type` `c` in `[__lo,__hi)` for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a matching `char_type` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### do\_scan\_not()

```
template<typename _CharT>
virtual const char_type * std::__ctype<_CharT>::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

|                        |                              |
|------------------------|------------------------------|
| <code>↔<br/>_m</code>  | The mask to compare against. |
| <code>↔<br/>_lo</code> | Pointer to start of range.   |
| <code>↔<br/>_hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a non-matching `char_type` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### `do_tolower()` [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_tolower (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

|                        |                            |
|------------------------|----------------------------|
| <code>↔<br/>_lo</code> | Pointer to start of range. |
| <code>↔<br/>_hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### `do_tolower()` [2/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_tolower (
    char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                                                                                         |                           |
|-----------------------------------------------------------------------------------------|---------------------------|
| <br>_c | The char_type to convert. |
|-----------------------------------------------------------------------------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else \_\_c.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::tolower\(\)](#), and [std::ctype<char>::tolower\(\)](#).

**do\_toupper()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<\_CharT>::do\_toupper (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                             |                            |
|---------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo  | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_toupper()** [2/2]

```
template<typename _CharT>
virtual char_type std::ctype<\_CharT>::do\_toupper (
    char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                           |                           |
|-------------------------------------------------------------------------------------------|---------------------------|
| <br>_c | The char_type to convert. |
|-------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by [std::ctype< char >::toupper\(\)](#), and [std::ctype< char >::toupper\(\)](#).

**do\_widen() [1/2]**

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_widen (
    char __c) const [protected], [virtual]
```

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                          |                      |
|------------------------------------------|----------------------|
| <a href="#">_↔</a><br><a href="#">_C</a> | The char to convert. |
|------------------------------------------|----------------------|

**Returns**

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by [std::ctype< char >::widen\(\)](#), and [std::ctype< char >::widen\(\)](#).

**do\_widen() [2/2]**

```
template<typename _CharT>
virtual const char * std::ctype< _CharT >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [protected], [virtual]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                           |                                   |
|-------------------------------------------|-----------------------------------|
| <a href="#">_↔</a><br><a href="#">_lo</a> | Pointer to start range.           |
| <a href="#">_↔</a><br><a href="#">_hi</a> | Pointer to end of range.          |
| <a href="#">_↔</a><br><a href="#">_to</a> | Pointer to the destination array. |

**Returns**

[\\_\\_hi](#).

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).



**is()** [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**is()** [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
    mask __m,
    char_type __c) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_↵type>::do\_is().

**Parameters**

|                        |                                       |
|------------------------|---------------------------------------|
| <code>↵<br/>__c</code> | The char_type to compare the mask of. |
| <code>↵<br/>__m</code> | The mask to compare against.          |

**Returns**

$(M \& \text{__m}) \neq 0$ .

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

**narrow()** [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
    char_type __c,
    char __dfault) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

## Parameters

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

## Returns

The converted char.

Referenced by `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_get<_CharT, _InIter>::get()`, and `std::time_put<_CharT, _OutIter>::put()`.

**narrow()** [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

## Returns

`__hi`.

**scan\_is()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__m</code> | The mask to compare against. |
|------------------|------------------------------|

**Parameters**

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

Pointer to matching `char_type` if found, else `__hi`.

**scan\_not()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Find `char_type` not matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is false. It does so by returning `ctype<char_type>::do_scan_not()`.

**Parameters**

|                                                                                                                          |                                 |
|--------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__m</code>   | The mask to compare against.    |
| <a href="#"></a><br><code>__lo</code> | Pointer to first char in range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**tolower() [1/2]**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
    char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

**Parameters**

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**`__hi`.**tolower()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::tolower (
    char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**The lowercase char\_type if convertible, else `__c`.Referenced by `std::time_get<_CharT, _InIter>::get()`.**toupper()** [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::toupper (
    char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**`__hi`.**toupper()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::toupper (
    char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

### Parameters

|                                                                                   |                           |
|-----------------------------------------------------------------------------------|---------------------------|
|  | The char_type to convert. |
| <code>__c</code>                                                                  |                           |

### Returns

The uppercase char\_type if convertible, else `__c`.

Referenced by [std::time\\_get<\\_CharT, \\_InIter >::get\(\)](#).

### widen() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT >::widen (
    char __c) const [inline], [inherited]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

### Parameters

|                                                                                   |                      |
|-----------------------------------------------------------------------------------|----------------------|
|  | The char to convert. |
| <code>__c</code>                                                                  |                      |

### Returns

The converted char\_type.

Referenced by [std::money\\_get<\\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter >::do\\_get\(\)](#), [std::money\\_put<\\_CharT, \\_OutIter >::do\\_put\(\)](#) and [std::time\\_put<\\_CharT, \\_OutIter >::do\\_put\(\)](#).

### widen() [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base<_CharT >::widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

### Parameters

|                                                                                     |                                   |
|-------------------------------------------------------------------------------------|-----------------------------------|
|  | Pointer to start of range.        |
| <code>__lo</code>                                                                   |                                   |
|  | Pointer to end of range.          |
| <code>__hi</code>                                                                   |                                   |
|  | Pointer to the destination array. |
| <code>__to</code>                                                                   |                                   |

Returns

`__hi`.

### 5.355.3 Member Data Documentation

**id**

```
template<typename _CharT>
locale::id std::ctype< _CharT >::id [static]
```

The facet id for ctype<char\_type>

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.356 std::ctype< char > Class Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype< char >:



### Public Types

- typedef const int \* `__to_type`
- typedef char `char_type`
- typedef char `char_type`
- typedef `__ctype_abstract_base< char >::mask` `mask`

## Public Member Functions

- `ctype` (`__c_locale __cloc`, `const mask * __table=0`, `bool __del=false`, `size_t __refs=0`)
- `ctype` (`const mask * __table=0`, `bool __del=false`, `size_t __refs=0`)
- `ctype` (`size_t __refs=0`)
- `const char_type * is` (`const char_type * __lo`, `const char_type * __hi`, `mask * __vec`) `const`
- `bool is` (`mask __m`, `char_type __c`) `const`
- `const char * is` (`const char * __lo`, `const char * __hi`, `mask * __vec`) `const`
- `const char_type * is` (`const char_type * __lo`, `const char_type * __hi`, `mask * __vec`) `const`
- `bool is` (`mask __m`, `char __c`) `const`
- `bool is` (`mask __m`, `char_type __c`) `const`
- `char narrow` (`char_type __c`, `char __default`) `const`
- `const char_type * narrow` (`const char_type * __lo`, `const char_type * __hi`, `char __default`, `char * __to`) `const`
- `char narrow` (`char_type __c`, `char __default`) `const`
- `char narrow` (`char_type __c`, `char __default`) `const`
- `const char_type * narrow` (`const char_type * __lo`, `const char_type * __hi`, `char __default`, `char * __to`) `const`
- `const char_type * narrow` (`const char_type * __lo`, `const char_type * __hi`, `char __default`, `char * __to`) `const`
- `const char_type * scan_is` (`mask __m`, `const char_type * __lo`, `const char_type * __hi`) `const`
- `const char * scan_is` (`mask __m`, `const char * __lo`, `const char * __hi`) `const`
- `const char_type * scan_is` (`mask __m`, `const char_type * __lo`, `const char_type * __hi`) `const`
- `const char_type * scan_not` (`mask __m`, `const char_type * __lo`, `const char_type * __hi`) `const`
- `const char * scan_not` (`mask __m`, `const char * __lo`, `const char * __hi`) `const`
- `const char_type * scan_not` (`mask __m`, `const char_type * __lo`, `const char_type * __hi`) `const`
- `const mask * table` () `const throw ()`
- `const char_type * tolower` (`char_type * __lo`, `const char_type * __hi`) `const`
- `char_type tolower` (`char_type __c`) `const`
- `const char_type * tolower` (`char_type * __lo`, `const char_type * __hi`) `const`
- `const char_type * tolower` (`char_type * __lo`, `const char_type * __hi`) `const`
- `char_type tolower` (`char_type __c`) `const`
- `char_type tolower` (`char_type __c`) `const`
- `const char_type * toupper` (`char_type * __lo`, `const char_type * __hi`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `const char_type * toupper` (`char_type * __lo`, `const char_type * __hi`) `const`
- `const char_type * toupper` (`char_type * __lo`, `const char_type * __hi`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `const char * widen` (`const char * __lo`, `const char * __hi`, `char_type * __to`) `const`
- `char_type widen` (`char __c`) `const`
- `char_type widen` (`char __c`) `const`
- `const char * widen` (`const char * __lo`, `const char * __hi`, `char_type * __to`) `const`
- `const char * widen` (`const char * __lo`, `const char * __hi`, `char_type * __to`) `const`

## Static Public Member Functions

- `static const mask * classic_table` () `throw ()`

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual [~ctype](#) ()
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const =0
- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const =0
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_default) const =0
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, [char](#) \_\_default, [char](#) \* \_\_to) const =0
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_default) const
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#), [char](#) \_\_default) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, [char](#) \_\_default, [char](#) \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, [char](#) \_\_default, [char](#) \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const =0
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const =0
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const =0
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const =0
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const =0
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const =0
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const [char](#) \* [do\\_widen](#) (const [char](#) \* \_\_lo, const [char](#) \* \_\_hi, [char\\_type](#) \* \_\_to) const =0
- virtual [char\\_type](#) [do\\_widen](#) ([char](#) \_\_c) const
- virtual [char\\_type](#) [do\\_widen](#) ([char](#) \_\_c) const
- virtual const [char](#) \* [do\\_widen](#) (const [char](#) \* \_\_lo, const [char](#) \* \_\_hi, [char\\_type](#) \* \_\_dest) const
- virtual const [char](#) \* [do\\_widen](#) (const [char](#) \* \_\_lo, const [char](#) \* \_\_hi, [char\\_type](#) \* \_\_to) const



## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Protected Attributes

- `__c_locale _M_c_locale_ctype`
- bool `_M_del`
- char `_M_narrow` [1+static\_cast< unsigned char >(-1)]
- char `_M_narrow_ok`
- const mask \* `_M_table`
- `__to_type _M_tolower`
- `__to_type _M_toupper`
- char `_M_widen` [1+static\_cast< unsigned char >(-1)]
- char `_M_widen_ok`

### 5.356.1 Detailed Description

The `ctype<char>` specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

### 5.356.2 Member Typedef Documentation

#### char\_type

```
typedef char std::ctype< char >::char_type
```

Typedef for the template parameter char.

### 5.356.3 Constructor & Destructor Documentation

#### ctype() [1/2]

```
std::ctype< char >::ctype (
    const mask * __table = 0,
    bool __del = false,
    size_t __refs = 0) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                      |                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------|
| <code>__table</code> | If non-zero, table is used as the per-char mask. Else <code>classic_table()</code> is used. |
| <code>__del</code>   | If true, passes ownership of table to this facet.                                           |
| <code>__refs</code>  | Passed to the base facet class.                                                             |

Referenced by `~ctype()`.

**ctype()** [2/2]

```
std::ctype< char >::ctype (
    __c_locale __cloc,
    const mask * __table = 0,
    bool __del = false,
    size_t __refs = 0) [explicit]
```

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__cloc</code>  | Handle to C locale data.                          |
| <code>__table</code> | If non-zero, table is used as the per-char mask.  |
| <code>__del</code>   | If true, passes ownership of table to this facet. |
| <code>__refs</code>  | Passed to the base facet class.                   |

**~ctype()**

```
virtual std::ctype< char >::~~ctype () [protected], [virtual]
```

Destructor.

This function deletes table() if *del* was true in the constructor.

References [ctype\(\)](#).

**5.356.4 Member Function Documentation****classic\_table()**

```
static const mask * std::ctype< char >::classic_table () throw ( ) [static]
```

Returns a pointer to the C locale mask table.

**do\_is()** [1/4]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [protected], [pure virtual], [inherited]
```

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), [std::ctype< wchar\\_t >](#), [std::mask< \\_CharT >](#), and [std::mask< \\_CharT >](#).

**do\_is()** [2/4]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base< _CharT >::do_is (
    mask __m,
    char_type __c) const [protected], [pure virtual], [inherited]
```

Test char\_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

**Returns**

$(M \& \text{__m}) \neq 0$ .

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), [std::ctype< wchar\\_t >](#), [std::mask< \\_CharT >](#), and [std::mask< \\_CharT >](#).

Referenced by [std::mask< \\_CharT >::is\(\)](#), and [std::mask< \\_CharT >::is\(\)](#).

**do\_is()** [3/4]

```
virtual const char_type * std::ctype< char >::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**do\_is()** [4/4]

```
virtual bool std::ctype< char >::do_is (
    mask __m,
    char_type __c) const [protected], [virtual]
```

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

|                          |                                    |
|--------------------------|------------------------------------|
| $\leftrightarrow$<br>__c | The char_type to find the mask of. |
| $\leftrightarrow$<br>__m | The mask to compare against.       |

**Returns**

(*M* & \_\_m) != 0.

**do\_narrow()** [1/6]

```
template<typename _CharT>
virtual char std::__ctype_abstract_base< _CharT >::do_narrow (
    char_type __c,
    char __default) const [protected], [pure virtual], [inherited]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, default is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|           |                                     |
|-----------|-------------------------------------|
| __c       | The char_type to convert.           |
| __default | Char to return if conversion fails. |

**Returns**

The converted char.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), [std::ctype< wchar\\_t >](#), [std::mask< \\_CharT >](#), and [std::mask< \\_CharT >](#).

Referenced by [std::mask< \\_CharT >::narrow\(\)](#), and [std::mask< \\_CharT >::narrow\(\)](#).

**do\_narrow()** [2/6]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
```

```
char __default,
char * __to) const [protected], [pure virtual], [inherited]
```

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

#### Returns

`__hi`.

Implemented in `std::ctype< _CharT >`, `std::ctype< wchar_t >`, `std::ctype< wchar_t >`, `std::mask< _CharT >`, and `std::mask< _CharT >`.

#### `do_narrow()` [3/6]

```
virtual char std::ctype< char >::do_narrow (
    char_type __c,
    char __default) const [inline], [protected], [virtual]
```

Narrow `char`.

This virtual function converts the `char` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an underived `ctype<char>` facet, `c` will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The <code>char</code> to convert.   |
| <code>__default</code> | Char to return if conversion fails. |

#### Returns

The converted `char`.

#### `do_narrow()` [4/6]

```
virtual char std::ctype< char >::do_narrow (
    char_type ,
    char __default) const [protected], [virtual]
```

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

**Returns**

The converted char.

**do\_narrow()** [5/6]

```
virtual const char_type * std::ctype< char >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [protected], [virtual]
```

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**do\_narrow()** [6/6]

```
virtual const char_type * std::ctype< char >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [inline], [protected], [virtual]
```

Narrow `char` array to `char` array.

This virtual function converts each `char` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char` in the input that cannot be converted, `default` is used instead. For an undervied `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

## Returns

`__hi`.

**do\_scan\_is()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [pure virtual], [inherited]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a matching char\_type if found, else `__hi`.

Implemented in `std::ctype< _CharT >`, `std::ctype< wchar_t >`, `std::ctype< wchar_t >`, `std::mask< _CharT >`, and `std::mask< _CharT >`.

Referenced by `std::mask< _CharT >::scan_is()`.

**do\_scan\_is()** [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.



**Parameters**

|                                                                                                                        |                              |
|------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching `char_type` if found, else `__hi`.

**do\_scan\_not()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [pure virtual], [inherited]
```

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

**Parameters**

|                                                                                                                          |                              |
|--------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching `char_type` if found, else `__hi`.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), [std::ctype< wchar\\_t >](#), [std::mask< \\_CharT >](#), and [std::mask< \\_CharT >](#).

Referenced by [std::mask< \\_CharT >::scan\\_not\(\)](#).

**do\_scan\_not()** [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

|                                                                                                                        |                              |
|------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a non-matching `char_type` if found, else `__hi`.

**do\_tolower()** [1/6]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_tolower (
    char_type * __lo,
    const char_type * __hi) const [protected], [pure virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

Implemented in `std::ctype< _CharT >`, `std::ctype< wchar_t >`, `std::ctype< wchar_t >`, `std::mask< _CharT >`, and `std::mask< _CharT >`.

**do\_tolower()** [2/6]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_tolower (
    char_type __c) const [protected], [pure virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

|                                                                                                                         |                                        |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"></a><br><code>__c</code> | The <code>char_type</code> to convert. |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Implemented in `std::ctype<_CharT>`, `std::ctype<wchar_t>`, `std::ctype<wchar_t>`, `std::mask<_CharT>`, and `std::mask<_CharT>`.

Referenced by `std::mask<_CharT>::tolower()`, and `std::mask<_CharT>::tolower()`.

**do\_tolower()** [3/6]

```
virtual const char_type * std::ctype< char >::do_tolower (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**do\_tolower()** [4/6]

```
virtual const char_type * std::ctype< char >::do_tolower (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each `char` in the range `[lo,hi)` to lowercase if possible. Other chars remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

**Returns**

`__hi`.

References `do_tolower()`.

**do\_tolower()** [5/6]

```
virtual char_type std::ctype< char >::do_tolower (
    char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else \_\_c.

**do\_tolower()** [6/6]

```
virtual char_type std::ctype< char >::do_tolower (
    char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The lowercase char if convertible, else \_\_c.

References [do\\_tolower\(\)](#).

Referenced by [do\\_tolower\(\)](#), and [do\\_tolower\(\)](#).

**do\_toupper()** [1/6]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_toupper (
    char_type * __lo,
    const char_type * __hi) const [protected], [pure virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), [std::ctype< wchar\\_t >](#), [std::mask< \\_CharT >](#), and [std::mask< \\_CharT >](#).

**do\_toupper()** [2/6]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_toupper (
    char_type __c) const [protected], [pure virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                                         |                                        |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"></a><br><code>__c</code> | The <code>char_type</code> to convert. |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), [std::ctype< wchar\\_t >](#), [std::mask< \\_CharT >](#), and [std::mask< \\_CharT >](#).

Referenced by [std::mask< \\_CharT >::toupper\(\)](#), and [std::mask< \\_CharT >::toupper\(\)](#).

**do\_toupper()** [3/6]

```
virtual const char_type * std::ctype< char >::do_toupper (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

\_\_hi.

**do\_toupper()** [4/6]

```
virtual const char_type * std::ctype< char >::do_toupper (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                           |                            |
|-------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

References [do\\_toupper\(\)](#).

**do\_toupper()** [5/6]

```
virtual char_type std::ctype< char >::do_toupper (
    char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                            |                           |
|--------------------------------------------------------------------------------------------|---------------------------|
| <br>__c | The char_type to convert. |
|--------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

**do\_toupper()** [6/6]

```
virtual char_type std::ctype< char >::do_toupper (
    char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                   |                      |
|---------------------------------------------------------------------------------------------------|----------------------|
|  <code>_c</code> | The char to convert. |
|---------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The uppercase char if convertible, else `_c`.

References [do\\_toupper\(\)](#).

Referenced by [do\\_toupper\(\)](#), and [do\\_toupper\(\)](#).

**do\_widen() [1/5]**

```
template<typename _CharT>
virtual const char * std::ctype\_abstract\_base< _CharT >::do_widen (
    const char * __lo,
    const char * __hi,
    char\_type * __to) const    [protected], [pure virtual], [inherited]
```

Widen char array.

This function converts each char in the input to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                                                                                                      |                                   |
|------------------------------------------------------------------------------------------------------|-----------------------------------|
|  <code>_lo</code> | Pointer to start range.           |
|  <code>_hi</code> | Pointer to end of range.          |
|  <code>_to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), [std::ctype< wchar\\_t >](#), [std::mask< \\_CharT >](#), and [std::mask< \\_CharT >](#).

**do\_widen() [2/5]**

```
virtual char\_type std::ctype< char >::do_widen (
    char __c) const    [protected], [virtual]
```

Widen char.

This virtual function converts the char to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                                                                                                     |                      |
|-----------------------------------------------------------------------------------------------------|----------------------|
|  <code>_c</code> | The char to convert. |
|-----------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**do\_widen()** [3/5]

```
virtual char_type std::ctype< char >::do_widen (
    char __c) const [inline], [protected], [virtual]
```

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                    |                      |
|----------------------------------------------------------------------------------------------------|----------------------|
|  <code>__c</code> | The char to convert. |
|----------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted character.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

References [do\\_widen\(\)](#).

Referenced by [do\\_widen\(\)](#).

**do\_widen()** [4/5]

```
virtual const char * std::ctype< char >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __dest) const [protected], [virtual]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                       |                                   |
|-------------------------------------------------------------------------------------------------------|-----------------------------------|
|  <code>__lo</code> | Pointer to start range.           |
|  <code>__hi</code> | Pointer to end of range.          |
|  <code>__to</code> | Pointer to the destination array. |

**Returns**

`__hi`.



**do\_widen()** [5/5]

```
virtual const char * std::ctype< char >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [inline], [protected], [virtual]
```

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an undervied ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

**is()** [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**is()** [2/6]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
    mask __m,
    char_type __c) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_↔type>::do\_is().

#### Parameters

|                          |                                       |
|--------------------------|---------------------------------------|
| <a href="#">↔</a><br>__c | The char_type to compare the mask of. |
| <a href="#">↔</a><br>__m | The mask to compare against.          |

#### Returns

(M & \_\_m) != 0.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

#### is() [3/6]

```
const char * std::ctype< char >::is (
    const char * __lo,
    const char * __hi,
    mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

#### Parameters

|       |                                      |
|-------|--------------------------------------|
| __lo  | Pointer to start of range.           |
| __hi  | Pointer to end of range.             |
| __vec | Pointer to an array of mask storage. |

#### Returns

\_\_hi.

#### is() [4/6]

```
const char_type * std::__ctype_abstract_base< char >::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

|       |                                      |
|-------|--------------------------------------|
| __lo  | Pointer to start of range.           |
| __hi  | Pointer to end of range.             |
| __vec | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**is()** [5/6]

```
bool std::ctype< char >::is (
    mask __m,
    char __c) const [inline]
```

Test char classification.

This function compares the mask table[c] to `__m`.

**Parameters**

|                  |                                  |
|------------------|----------------------------------|
| <code>__c</code> | The char to compare the mask of. |
| <code>__m</code> | The mask to compare against.     |

**Returns**

True if `__m & table[__c]` is true, false otherwise.

**is()** [6/6]

```
bool std::__ctype_abstract_base< char >::is (
    mask __m,
    char_type __c) const [inline]
```

Test `char_type` classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

**Parameters**

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__c</code> | The <code>char_type</code> to compare the mask of. |
| <code>__m</code> | The mask to compare against.                       |

**Returns**

`(M & __m) != 0`.

**narrow()** [1/6]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
    char_type __c,
    char __default) const [inline], [inherited]
```

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `default` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

## Returns

The converted char.

Referenced by `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_get<_CharT, _InIter>::get()`, and `std::time_put<_CharT, _OutIter>::put()`.

**narrow()** [2/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

## Returns

`__hi`.

**narrow()** [3/6]

```
char std::__ctype_abstract_base< char >::narrow (
    char_type __c,
    char __default) const [inline]
```

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, *default* is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

**Returns**

The converted char.

**narrow()** [4/6]

```
char std::ctype< char >::narrow (
    char_type __c,
    char __default) const [inline]
```

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, *default* is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.

This function works as if it returns ctype<char>::do\_narrow(*c*). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char to convert.                |
| <code>__default</code> | Char to return if conversion fails. |

**Returns**

The converted character.

References [std::ctype<\\_CharT>::do\\_narrow\(\)](#).

**narrow()** [5/6]

```
const char_type * std::__ctype_abstract_base< char >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [inline]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, *default* is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_default, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**narrow()** [6/6]

```
const char_type * std::ctype< char >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [inline]
```

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_narrow(lo, hi, default, to). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

References [std::ctype< \\_CharT >::do\\_narrow\(\)](#).

**scan\_is()** [1/3]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else `__hi`.

**scan\_is()** [2/3]

```
const char * std::ctype< char >::scan_is (
    mask __m,
    const char * __lo,
    const char * __hi) const [inline]
```

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching char if found, else `__hi`.

**scan\_is()** [3/3]

```
const char_type * std::__ctype_abstract_base< char >::scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else `__hi`.

**scan\_not()** [1/3]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is false. It does so by returning `ctype<char_type>::do_scan_not()`.



**Parameters**

|                                                                                                                        |                                 |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against.    |
| <a href="#"></a><br><code>__lo</code> | Pointer to first char in range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**scan\_not() [2/3]**

```
const char * std::ctype< char >::scan_not (
    mask __m,
    const char * __lo,
    const char * __hi) const [inline]
```

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [`__lo`,`__hi`) for which `is(m,char)` is false.

**Parameters**

|                                                                                                                          |                              |
|--------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>    | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char if found, else `__hi`.

**scan\_not() [3/3]**

```
const char\_type * std::\_\_ctype\_abstract\_base< char >::scan_not (
    mask __m,
    const char\_type * __lo,
    const char\_type * __hi) const [inline]
```

Find `char_type` not matching a mask.

This function searches for and returns the first `char_type` c in [`lo`,`hi`) for which `is(m,c)` is false. It does so by returning `ctype<char_type>::do_scan_not()`.

**Parameters**

|                                                                                                                          |                                 |
|--------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against.    |
| <a href="#"></a><br><code>__lo</code> | Pointer to first char in range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**table()**

```
const mask * std::ctype< char >::table () const throw ( ) [inline]
```

Returns a pointer to the mask table provided to the constructor, or the default from `classic_table()` if none was provided.

**tolower() [1/6]**

```
template<typename _CharT>
```

```
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
    char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**tolower() [2/6]**

```
template<typename _CharT>
```

```
char_type std::__ctype_abstract_base< _CharT >::tolower (
    char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__c</code> | The <code>char_type</code> to convert. |
|------------------|----------------------------------------|

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Referenced by `std::time_get< _CharT, _InIter >::get()`.

**tolower()** [3/6]

```
const char_type * std::__ctype_abstract_base< char >::tolower (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

|                      |                            |
|----------------------|----------------------------|
| $\leftarrow$<br>__lo | Pointer to start of range. |
| $\leftarrow$<br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

**tolower()** [4/6]

```
const char_type * std::ctype< char >::tolower (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

tolower() acts as if it returns ctype<char>::do\_tolower(\_\_lo, \_\_hi). do\_tolower() must always return the same result for the same input.

**Parameters**

|                      |                                 |
|----------------------|---------------------------------|
| $\leftarrow$<br>__lo | Pointer to first char in range. |
| $\leftarrow$<br>__hi | Pointer to end of range.        |

**Returns**

\_\_hi.

References [std::ctype<\\_CharT>::do\\_tolower\(\)](#).

**tolower()** [5/6]

```
char_type std::__ctype_abstract_base< char >::tolower (
    char_type __c) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

## Parameters

|                                    |                                        |
|------------------------------------|----------------------------------------|
| <code>↔</code><br><code>__c</code> | The <code>char_type</code> to convert. |
|------------------------------------|----------------------------------------|

## Returns

The lowercase `char_type` if convertible, else `__c`.

**tolower()** [6/6]

```
char_type std::ctype< char >::tolower (
    char_type __c) const [inline]
```

Convert to lowercase.

This function converts the `char` argument to lowercase if possible. If not possible (for example, '2'), returns the argument. `tolower()` acts as if it returns `ctype<char>::do_tolower(__c)`. `do_tolower()` must always return the same result for the same input.

## Parameters

|                                    |                                   |
|------------------------------------|-----------------------------------|
| <code>↔</code><br><code>__c</code> | The <code>char</code> to convert. |
|------------------------------------|-----------------------------------|

## Returns

The lowercase `char` if convertible, else `__c`.

References [std::ctype< \\_CharT >::do\\_tolower\(\)](#).

**toupper()** [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
    char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

## Parameters

|                                     |                            |
|-------------------------------------|----------------------------|
| <code>↔</code><br><code>__lo</code> | Pointer to start of range. |
| <code>↔</code><br><code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

**toupper()** [2/6]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
    char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

**Parameters**

|                          |                           |
|--------------------------|---------------------------|
| $\leftrightarrow$<br>__c | The char_type to convert. |
|--------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

Referenced by [std::time\\_get< \\_CharT, \\_Inlter >::get\(\)](#).

**toupper()** [3/6]

```
const char_type * std::__ctype_abstract_base< char >::toupper (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

**Parameters**

|                           |                            |
|---------------------------|----------------------------|
| $\leftrightarrow$<br>__lo | Pointer to start of range. |
| $\leftrightarrow$<br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

**toupper()** [4/6]

```
const char_type * std::ctype< char >::toupper (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char in the range [\_\_lo,\_\_hi) to uppercase if possible. Other chars remain untouched.

toupper() acts as if it returns ctype<char>::do\_toupper(\_\_lo, \_\_hi). do\_toupper() must always return the same result for the same input.

**Parameters**

|                           |                                 |
|---------------------------|---------------------------------|
| $\leftrightarrow$<br>__lo | Pointer to first char in range. |
|---------------------------|---------------------------------|

|                                                                                                   |                          |
|---------------------------------------------------------------------------------------------------|--------------------------|
| <a href="#"></a> | Pointer to end of range. |
| <a href="#"></a> |                          |
| <a href="#"></a> |                          |
| <a href="#"></a> |                          |

**Returns**[!\[\]\(d3fb9f94af8b26d1c844efa9a98805b0\_img.jpg\)](#) *hi*.References [std::ctype< \\_CharT >::do\\_toupper\(\)](#).**toupper()** [5/6]

```
char_type std::__ctype_abstract_base< char >::toupper (
    char_type __c) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

**Parameters**

|                                                                                                   |                           |
|---------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a> | The char_type to convert. |
| <a href="#"></a> |                           |
| <a href="#"></a> |                           |
| <a href="#"></a> |                           |

**Returns**The uppercase char\_type if convertible, else `__c`.**toupper()** [6/6]

```
char_type std::ctype< char >::toupper (
    char_type __c) const [inline]
```

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument. `toupper()` acts as if it returns `ctype<char>::do_toupper(c)`. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                     |                      |
|-----------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a> | The char to convert. |
| <a href="#"></a> |                      |
| <a href="#"></a> |                      |
| <a href="#"></a> |                      |

**Returns**The uppercase char if convertible, else `__c`.References [std::ctype< \\_CharT >::do\\_toupper\(\)](#).**widen()** [1/5]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [inline], [inherited]
```

Widen array to `char_type`.

This function converts each char in the input to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

#### Returns

`__hi`.

#### **widen()** [2/5]

```
char_type std::__ctype_abstract_base< char >::widen (  
    char __c) const [inline]
```

Widen char to `char_type`.

This function converts the char argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

#### Returns

The converted `char_type`.

#### **widen()** [3/5]

```
char_type std::ctype< char >::widen (  
    char __c) const [inline]
```

Widen char.

This function converts the char to `char_type` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The converted character.

References [std::ctype< \\_CharT >::do\\_widen\(\)](#).

**widen()** [4/5]

```
const char * std::__ctype_abstract_base< char >::widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [inline]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                                     |                                   |
|---------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#"> <code>__lo</code></a> | Pointer to start of range.        |
| <a href="#"> <code>__hi</code></a> | Pointer to end of range.          |
| <a href="#"> <code>__to</code></a> | Pointer to the destination array. |

**Returns**

`__hi`.

**widen()** [5/5]

```
const char * std::ctype< char >::widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [inline]
```

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_widen(c). do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                                       |                                   |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#"> <code>__lo</code></a> | Pointer to first char in range.   |
| <a href="#"> <code>__hi</code></a> | Pointer to end of range.          |
| <a href="#"> <code>__to</code></a> | Pointer to the destination array. |

**Returns**

`__hi`.

References [std::ctype< \\_CharT >::do\\_widen\(\)](#).



### 5.356.5 Member Data Documentation

**id** [1/2]

```
locale::id std::ctype< char >::id [static]
```

The facet id for ctype<char\_type>

**id** [2/2]

```
locale::id std::ctype< char >::id [static]
```

The facet id for ctype<char>

**table\_size**

```
const size_t std::ctype< char >::table_size [static]
```

The size of the mask table. It is SCHAR\_MAX + 1.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [ctype\\_inline.h](#)

### 5.357 std::ctype< wchar\_t > Class Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype< wchar\_t >:



#### Public Types

- typedef const int \* **\_\_to\_type**
- typedef wctype\_t **\_\_wmask\_type**
- typedef wchar\_t **char\_type**
- typedef wchar\_t [char\\_type](#)
- typedef [\\_\\_ctype\\_abstract\\_base](#)< wchar\_t >::mask **mask**

**Public Member Functions**

- [ctype](#) ([\\_\\_c\\_locale](#) [\\_\\_cloc](#), [size\\_t](#) [\\_\\_refs](#)=0)
- [ctype](#) ([size\\_t](#) [\\_\\_refs](#)=0)
- [ctype](#) ([size\\_t](#) [\\_\\_refs](#)=0)
- const [char\\_type](#) \* [is](#) (const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#), [mask](#) \* [\\_\\_vec](#)) const
- bool [is](#) ([mask](#) [\\_\\_m](#), [char\\_type](#) [\\_\\_c](#)) const
- [char](#) [narrow](#) ([char\\_type](#) [\\_\\_c](#), [char](#) [\\_\\_default](#)) const
- const [char\\_type](#) \* [narrow](#) (const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#), [char](#) [\\_\\_default](#), [char](#) \* [\\_\\_to](#)) const
- const [char\\_type](#) \* [scan\\_is](#) ([mask](#) [\\_\\_m](#), const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#)) const
- const [char\\_type](#) \* [scan\\_not](#) ([mask](#) [\\_\\_m](#), const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#)) const
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#)) const
- [char\\_type](#) [tolower](#) ([char\\_type](#) [\\_\\_c](#)) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#)) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) [\\_\\_c](#)) const
- [char\\_type](#) [widen](#) ([char](#) [\\_\\_c](#)) const
- const [char](#) \* [widen](#) (const [char](#) \* [\\_\\_lo](#), const [char](#) \* [\\_\\_hi](#), [char\\_type](#) \* [\\_\\_to](#)) const

**Static Public Attributes**

- static const [mask](#) [alnum](#)
- static const [mask](#) [alpha](#)
- static const [mask](#) [blank](#)
- static const [mask](#) [cntrl](#)
- static const [mask](#) [digit](#)
- static const [mask](#) [graph](#)
- static [locale::id](#) [id](#)
- static [locale::id](#) [id](#)
- static const [mask](#) [lower](#)
- static const [mask](#) [print](#)
- static const [mask](#) [punct](#)
- static const [mask](#) [space](#)
- static const [mask](#) [upper](#)
- static const [mask](#) [xdigit](#)

**Protected Member Functions**

- virtual [~ctype](#) ()
- [\\_\\_wmask\\_type](#) [\\_M\\_convert\\_to\\_wmask](#) (const [mask](#) [\\_\\_m](#)) const throw ()
- void [\\_M\\_initialize\\_ctype](#) () throw ()
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#), [mask](#) \* [\\_\\_vec](#)) const
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#), [mask](#) \* [\\_\\_vec](#)) const
- virtual bool [do\\_is](#) ([mask](#) [\\_\\_m](#), [char\\_type](#) [\\_\\_c](#)) const
- virtual bool [do\\_is](#) ([mask](#) [\\_\\_m](#), [char\\_type](#) [\\_\\_c](#)) const
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#) [\\_\\_c](#), [char](#) [\\_\\_default](#)) const
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#), [char](#) [\\_\\_default](#)) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#), [char](#) [\\_\\_default](#), [char](#) \* [\\_\\_to](#)) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#), [char](#) [\\_\\_default](#), [char](#) \* [\\_\\_to](#)) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) ([mask](#) [\\_\\_m](#), const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#)) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) ([mask](#) [\\_\\_m](#), const [char\\_type](#) \* [\\_\\_lo](#), const [char\\_type](#) \* [\\_\\_hi](#)) const

- virtual const `char_type * do_scan_not` (mask \_\_m, const `char_type * __lo`, const `char_type * __hi`) const
- virtual const `char_type * do_scan_not` (mask \_\_m, const `char_type * __lo`, const `char_type * __hi`) const
- virtual const `char_type * do_tolower` (`char_type * __lo`, const `char_type * __hi`) const
- virtual const `char_type * do_tolower` (`char_type * __lo`, const `char_type * __hi`) const
- virtual `char_type do_tolower` (`char_type __c`) const
- virtual `char_type do_tolower` (`char_type __c`) const
- virtual const `char_type * do_toupper` (`char_type * __lo`, const `char_type * __hi`) const
- virtual const `char_type * do_toupper` (`char_type * __lo`, const `char_type * __hi`) const
- virtual `char_type do_toupper` (`char_type __c`) const
- virtual `char_type do_toupper` (`char_type __c`) const
- virtual `char_type do_widen` (`char __c`) const
- virtual `char_type do_widen` (`char __c`) const
- virtual const `char * do_widen` (const `char * __lo`, const `char * __hi`, `char_type * __dest`) const
- virtual const `char * do_widen` (const `char * __lo`, const `char * __hi`, `char_type * __to`) const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale & __cloc`) throw ()
- static void `_S_create_c_locale` (`__c_locale & __cloc`, const `char * __s`, `__c_locale __old=0`)
- static void `_S_destroy_c_locale` (`__c_locale & __cloc`)
- static `__c_locale _S_get_c_locale` ()
- static const `char * _S_get_c_name` () throw ()
- static `__c_locale _S_lc_type_c_locale` (`__c_locale __cloc`, const `char * __s`)

### Protected Attributes

- mask `_M_bit` [16]
- `__c_locale _M_c_locale_ctype`
- `char _M_narrow` [128]
- bool `_M_narrow_ok`
- `wint_t _M_widen` [1+static\_cast< unsigned char >(-1)]
- `__wmask_type _M_wmask` [16]

#### 5.357.1 Detailed Description

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well. `ctype<wchar_t>` inherits its public methods from `__ctype_abstract_base<wchar_t>`.

#### 5.357.2 Member Typedef Documentation

##### `char_type`

```
typedef wchar_t std::ctype< wchar_t >::char_type
```

Typedef for the template parameter `wchar_t`.

#### 5.357.3 Constructor & Destructor Documentation

##### `ctype()` [1/2]

```
std::ctype< wchar_t >::ctype (
    size_t __refs = 0) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

**ctype()** [2/2]

```
std::ctype< wchar_t >::ctype (
    __c_locale __cloc,
    size_t __refs = 0) [explicit]
```

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | Handle to C locale data.        |
| <code>__refs</code> | Passed to the base facet class. |

**~ctype()**

```
virtual std::ctype< wchar_t >::~~ctype () [protected], [virtual]
```

Destructor.

**5.357.4 Member Function Documentation****do\_is()** [1/4]

```
virtual const char_type * std::ctype< wchar_t >::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

## Returns

`__hi`.

Implements `std::__ctype_abstract_base< wchar_t >`.

**do\_is()** [2/4]

```
virtual const char_type * std::ctype< wchar_t >::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each `wchar_t` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

Implements `std::__ctype_abstract_base< wchar_t >`.

**do\_is()** [3/4]

```
virtual bool std::ctype< wchar_t >::do_is (
    mask __m,
    char_type __c) const [protected], [virtual]
```

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__c</code> | The <code>char_type</code> to find the mask of. |
| <code>__m</code> | The mask to compare against.                    |

**Returns**

`(M & __m) != 0`.

Implements `std::__ctype_abstract_base< wchar_t >`.

**do\_is()** [4/4]

```
virtual bool std::ctype< wchar_t >::do_is (
    mask __m,
    char_type __c) const [protected], [virtual]
```

Test `wchar_t` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

## Parameters

|                                   |                                               |
|-----------------------------------|-----------------------------------------------|
| <a href="#"><code>_↔_c</code></a> | The <code>wchar_t</code> to find the mask of. |
| <a href="#"><code>_↔_m</code></a> | The mask to compare against.                  |

## Returns

$(M \& \_m) \neq 0$ .

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_narrow()** [1/4]

```
virtual char std::ctype< wchar_t >::do_narrow (
    char_type __c,
    char __dfault) const [protected], [virtual]
```

Narrow `wchar_t` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an underived `ctype<wchar_t>` facet, `c` will be cast to `char` and returned.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                                       |                                      |
|---------------------------------------|--------------------------------------|
| <a href="#"><code>__c</code></a>      | The <code>wchar_t</code> to convert. |
| <a href="#"><code>__dfault</code></a> | Char to return if conversion fails.  |

## Returns

The converted `char`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_narrow()** [2/4]

```
virtual char std::ctype< wchar_t >::do_narrow (
    char_type ,
    char __dfault) const [protected], [virtual]
```

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                                       |                                        |
|---------------------------------------|----------------------------------------|
| <a href="#"><code>__c</code></a>      | The <code>char_type</code> to convert. |
| <a href="#"><code>__dfault</code></a> | Char to return if conversion fails.    |

## Returns

The converted `char`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_narrow()** [3/4]

```
virtual const char_type * std::ctype< wchar_t >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [protected], [virtual]
```

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

Implements `std::__ctype_abstract_base< wchar_t >`.

**do\_narrow()** [4/4]

```
virtual const char_type * std::ctype< wchar_t >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [protected], [virtual]
```

Narrow wchar\_t array to char array.

This virtual function converts each wchar\_t in the range [`lo`,`hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any wchar\_t in the input that cannot be converted, `default` is used instead. For an undervied `ctype<wchar_t>` facet, the argument will be copied, casting each element to char.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

Implements `std::__ctype_abstract_base< wchar_t >`.

**do\_scan\_is()** [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching char\_type if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_scan\_is()** [2/2]

```
virtual const char_type * std::ctype< wchar_t >::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Find wchar\_t matching mask.

This function searches for and returns the first wchar\_t c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching wchar\_t if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).



**do\_scan\_not()** [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else `__hi`.

Implements `std::__ctype_abstract_base< wchar_t >`.

**do\_scan\_not()** [2/2]

```
virtual const char_type * std::ctype< wchar_t >::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Find wchar\_t not matching mask.

This function searches for and returns a pointer to the first wchar\_t c of [\_\_lo,\_\_hi) for which is(\_\_m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching wchar\_t if found, else `__hi`.

Implements `std::__ctype_abstract_base< wchar_t >`.

**do\_tolower()** [1/4]

```
virtual const char_type * std::ctype< wchar_t >::do_tolower (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                           |                            |
|-------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_tolower()** [2/4]

```
virtual const char_type * std::ctype< wchar_t >::do_tolower (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each wchar\_t in the range [lo,hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                             |                            |
|---------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_tolower()** [3/4]

```
virtual char_type std::ctype< wchar_t >::do_tolower (
    char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                                   |                           |
|---------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a> | The char_type to convert. |
| <code>__c</code>                                                                                  |                           |

**Returns**

The lowercase char\_type if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_tolower()** [4/4]

```
virtual char_type std::ctype< wchar_t >::do_tolower (  
    char_type __c) const    [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                                   |                         |
|---------------------------------------------------------------------------------------------------|-------------------------|
| <a href="#"></a> | The wchar_t to convert. |
| <code>__c</code>                                                                                  |                         |

**Returns**

The lowercase wchar\_t if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_toupper()** [1/4]

```
virtual const char_type * std::ctype< wchar_t >::do_toupper (  
    char_type * __lo,  
    const char_type * __hi) const    [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                     |                            |
|-----------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a> | Pointer to start of range. |
| <code>__lo</code>                                                                                   |                            |
| <a href="#"></a> | Pointer to end of range.   |
| <code>__hi</code>                                                                                   |                            |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_toupper()** [2/4]

```
virtual const char_type * std::ctype< wchar_t >::do_toupper (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each wchar\_t in the range [lo,hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                           |                            |
|-------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_toupper()** [3/4]

```
virtual char_type std::ctype< wchar_t >::do_toupper (
    char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                            |                           |
|--------------------------------------------------------------------------------------------|---------------------------|
| <br>__c | The char_type to convert. |
|--------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_toupper()** [4/4]

```
virtual char_type std::ctype< wchar_t >::do_toupper (
    char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the wchar\_t argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                   |                         |
|-----------------------------------------------------------------------------------|-------------------------|
|  | The wchar_t to convert. |
| <code>__c</code>                                                                  |                         |

**Returns**

The uppercase wchar\_t if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_widen() [1/4]**

```
virtual char_type std::ctype< wchar_t >::do_widen (  
    char __c) const    [protected], [virtual]
```

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                                                                                   |                      |
|-----------------------------------------------------------------------------------|----------------------|
|  | The char to convert. |
| <code>__c</code>                                                                  |                      |

**Returns**

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_widen() [2/4]**

```
virtual char_type std::ctype< wchar_t >::do_widen (  
    char __c) const    [protected], [virtual]
```

Widen char to wchar\_t.

This virtual function converts the char to wchar\_t using the simplest reasonable transformation. For an underived ctype<wchar\_t> facet, the argument will be cast to wchar\_t.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                                                                                     |                      |
|-------------------------------------------------------------------------------------|----------------------|
|  | The char to convert. |
| <code>__c</code>                                                                    |                      |

**Returns**

The converted wchar\_t.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_widen()** [3/4]

```
virtual const char * std::ctype< wchar_t >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __dest) const [protected], [virtual]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                     |                                   |
|-----------------------------------------------------------------------------------------------------|-----------------------------------|
|  <code>__lo</code> | Pointer to start range.           |
|  <code>__hi</code> | Pointer to end of range.          |
|  <code>__to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_widen()** [4/4]

```
virtual const char * std::ctype< wchar_t >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [protected], [virtual]
```

Widen char array to wchar\_t array.

This function converts each char in the input to wchar\_t using the simplest reasonable transformation. For an underived ctype<wchar\_t> facet, the argument will be copied, casting each element to wchar\_t.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                       |                                   |
|-------------------------------------------------------------------------------------------------------|-----------------------------------|
|  <code>__lo</code> | Pointer to start range.           |
|  <code>__hi</code> | Pointer to end of range.          |
|  <code>__to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**is()** [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**is()** [2/2]

```
bool std::__ctype_abstract_base< wchar_t >::is (
    mask __m,
    char_type __c) const [inline]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The char_type to compare the mask of. |
| <code>__m</code> | The mask to compare against.          |

**Returns**

$(M \& \text{__m}) \neq 0$ .

**narrow()** [1/2]

```
char std::__ctype_abstract_base< wchar_t >::narrow (
    char_type __C,
    char __default) const [inline]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__default</code> | Char to return if conversion fails. |
|------------------------|-------------------------------------|

**Returns**

The converted char.

**narrow()** [2/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to) const [inline]
```

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecv` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**scan\_is()**

```
const char_type * std::__ctype_abstract_base< wchar_t >::scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline]
```

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching `char_type` if found, else `__hi`.



**scan\_not()**

```
const char_type * std::__ctype_abstract_base< wchar_t >::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__m</code>  | The mask to compare against.    |
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**tolower() [1/2]**

```
const char_type * std::__ctype_abstract_base< wchar_t >::tolower (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**tolower() [2/2]**

```
char_type std::__ctype_abstract_base< wchar_t >::tolower (
    char_type __c) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

## Parameters

|                         |                           |
|-------------------------|---------------------------|
| $\leftrightarrow$<br>_c | The char_type to convert. |
|-------------------------|---------------------------|

## Returns

The lowercase char\_type if convertible, else \_\_c.

**toupper()** [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::toupper (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

## Parameters

|                          |                            |
|--------------------------|----------------------------|
| $\leftrightarrow$<br>_lo | Pointer to start of range. |
| $\leftrightarrow$<br>_hi | Pointer to end of range.   |

## Returns

\_\_hi.

**toupper()** [2/2]

```
char_type std::__ctype_abstract_base< wchar_t >::toupper (
    char_type __c) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

## Parameters

|                         |                           |
|-------------------------|---------------------------|
| $\leftrightarrow$<br>_c | The char_type to convert. |
|-------------------------|---------------------------|

## Returns

The uppercase char\_type if convertible, else \_\_c.

**widen()** [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::widen (
    char __c) const [inline]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                                                |                      |
|--------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><a href="#"><u>_c</u></a> | The char to convert. |
|--------------------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted char\_type.

**widen() [2/2]**

```
const char * std::\_\_ctype\_abstract\_base< wchar_t >::widen (  
    const char * \_\_lo,  
    const char * \_\_hi,  
    char\_type * \_\_to) const [inline]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                                                   |                                   |
|-----------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#"></a><br><a href="#"><u>_lo</u></a>   | Pointer to start of range.        |
| <a href="#"></a><br><a href="#"><u>_hi</u></a>   | Pointer to end of range.          |
| <a href="#"></a><br><a href="#"><u>_to</u></a> | Pointer to the destination array. |

**Returns**

[\\_\\_hi](#).

**5.357.5 Member Data Documentation****id [1/2]**

```
locale::id std::ctype< wchar_t >::id [static]
```

The facet id for ctype<char\_type>

**id [2/2]**

```
locale::id std::ctype< wchar_t >::id [static]
```

The facet id for ctype<wchar\_t>

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

**5.358 std::ctype\_base Struct Reference**

```
#include <ctype_base.h>
```

Inheritance diagram for std::ctype\_base:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef unsigned short **mask**

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### 5.358.1 Detailed Description

Base class for ctype.

The documentation for this struct was generated from the following file:

- [ctype\\_base.h](#)

## 5.359 std::ctype\_byname<\_CharT> Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for `std::ctype_byname<_CharT>`:



### Public Types

- `typedef const int * __to_type`
- `typedef _CharT char_type`
- `typedef ctype<_CharT>::mask mask`

### Public Member Functions

- `ctype_byname` (const char \*\_\_s, size\_t \_\_refs=0)
- `ctype_byname` (const string &\_\_s, size\_t \_\_refs=0)
- `const char_type * is` (const char\_type \*\_\_lo, const char\_type \*\_\_hi, mask \*\_\_vec) const
- `bool is` (mask \_\_m, char\_type \_\_c) const
- `char narrow` (char\_type \_\_c, char \_\_dfault) const
- `const char_type * narrow` (const char\_type \*\_\_lo, const char\_type \*\_\_hi, char \_\_dfault, char \*\_\_to) const
- `const char_type * scan_is` (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- `const char_type * scan_not` (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- `const char_type * tolower` (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- `char_type tolower` (char\_type \_\_c) const
- `const char_type * toupper` (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- `char_type toupper` (char\_type \_\_c) const
- `char_type widen` (char \_\_c) const
- `const char * widen` (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_to) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual const char\_type \* [do\\_is](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- virtual bool [do\\_is](#) (mask \_\_m, char\_type \_\_c) const
- virtual char [do\\_narrow](#) (char\_type, char \_\_dfault) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual const char\_type \* [do\\_scan\\_is](#) (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* [do\\_scan\\_not](#) (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_dest) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

#### 5.359.1 Detailed Description

```
template<typename _CharT>
class std::ctype_byname<_CharT>
```

class ctype\_byname [22.2.1.2].

## 5.359.2 Member Function Documentation

### do\_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [protected], [virtual], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

### do\_is() [2/2]

```
template<typename _CharT>
virtual bool std::ctype< _CharT >::do_is (
    mask __m,
    char_type __c) const [protected], [virtual], [inherited]
```

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

#### Returns

$(M \& \_m) \neq 0$ .

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

### do\_narrow() [1/2]

```
template<typename _CharT>
virtual char std::ctype< _CharT >::do_narrow (
```

```
char_type __c,
char __dfault) const [protected], [virtual], [inherited]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char_type to convert.           |
| <code>__dfault</code> | Char to return if conversion fails. |

#### Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::narrow\(\)](#), and [std::ctype<char>::narrow\(\)](#).

#### do\_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __dfault,
    char * __to) const [protected], [virtual], [inherited]
```

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### do\_scan\_is()

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_scan_is (
```



```

mask __m,
const char_type * __lo,
const char_type * __hi) const [protected], [virtual], [inherited]

```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [\_\_lo,\_\_hi) for which is(\_\_m,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>↔<br/>__m</code>  | The mask to compare against. |
| <code>↔<br/>__lo</code> | Pointer to start of range.   |
| <code>↔<br/>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a matching char\_type if found, else \_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### do\_scan\_not()

```

template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual], [inherited]

```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>↔<br/>__m</code>  | The mask to compare against. |
| <code>↔<br/>__lo</code> | Pointer to start of range.   |
| <code>↔<br/>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a non-matching char\_type if found, else \_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### do\_tolower() [1/2]

```

template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_tolower (

```

```
char_type * __lo,
const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                                                                                                     |                            |
|-----------------------------------------------------------------------------------------------------|----------------------------|
|  <code>__lo</code> | Pointer to start of range. |
|  <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### do\_tolower() [2/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_tolower (
    char_type __c) const [protected], [virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                                                                                                      |                           |
|------------------------------------------------------------------------------------------------------|---------------------------|
|  <code>__c</code> | The char_type to convert. |
|------------------------------------------------------------------------------------------------------|---------------------------|

#### Returns

The lowercase char\_type if convertible, else \_\_c.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::tolower\(\)](#), and [std::ctype<char>::tolower\(\)](#).

#### do\_toupper() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_toupper (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                                       |                            |
|-----------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>_lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>_hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_toupper()** [2/2]

```
template<typename _CharT>
virtual char_type std::ctype<\_CharT>::do\_toupper (
    char_type __c) const [protected], [virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                                       |                                        |
|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"></a><br><code>__c</code> | The <code>char_type</code> to convert. |
|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------|

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::toupper\(\)](#), and [std::ctype<char>::toupper\(\)](#).

**do\_widen()** [1/2]

```
template<typename _CharT>
virtual char_type std::ctype<\_CharT>::do\_widen (
    char __c) const [protected], [virtual], [inherited]
```

Widen char.

This virtual function converts the `char` to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                                                                                                                         |                      |
|-------------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__c</code> | The char to convert. |
|-------------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted `char_type`

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::widen\(\)](#), and [std::ctype<char>::widen\(\)](#).

**do\_widen()** [2/2]

```
template<typename _CharT>
virtual const char * std::ctype<_CharT>::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [protected], [virtual], [inherited]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                   |                                   |
|-----------------------------------|-----------------------------------|
| <a href="#"><code>__lo</code></a> | Pointer to start range.           |
| <a href="#"><code>__hi</code></a> | Pointer to end of range.          |
| <a href="#"><code>__to</code></a> | Pointer to the destination array. |

**Returns**

[`\_\_hi`](#).

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**is()** [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <a href="#"><code>__lo</code></a>  | Pointer to start of range.           |
| <a href="#"><code>__hi</code></a>  | Pointer to end of range.             |
| <a href="#"><code>__vec</code></a> | Pointer to an array of mask storage. |

**Returns**

[`\_\_hi`](#).

**is()** [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base<_CharT>::is (
```

```
mask __m,
char_type __c) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_↔type>::do\_is().

#### Parameters

|                          |                                       |
|--------------------------|---------------------------------------|
| <a href="#">↔</a><br>__c | The char_type to compare the mask of. |
| <a href="#">↔</a><br>__m | The mask to compare against.          |

#### Returns

(M & \_\_m) != 0.

Referenced by [std::time\\_get<\\_CharT, \\_InIter >::get\(\)](#).

#### narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base<_CharT >::narrow (
    char_type __c,
    char __dfault) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|          |                                     |
|----------|-------------------------------------|
| __c      | The char_type to convert.           |
| __dfault | Char to return if conversion fails. |

#### Returns

The converted char.

Referenced by [std::time\\_get<\\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::time\\_get<\\_CharT, \\_InIter >::get\(\)](#), and [std::time\\_put<\\_CharT, \\_OutIter >::put\(\)](#).

#### narrow() [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __dfault,
    char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_dfault, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

## Returns

`__hi`.

**scan\_is()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

## Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>↔<br/>__m</code>  | The mask to compare against. |
| <code>↔<br/>__lo</code> | Pointer to start of range.   |
| <code>↔<br/>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to matching char\_type if found, else `__hi`.

**scan\_not()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

## Parameters

|                         |                                 |
|-------------------------|---------------------------------|
| <code>↔<br/>__m</code>  | The mask to compare against.    |
| <code>↔<br/>__lo</code> | Pointer to first char in range. |

|                                                                                                                                  |                          |
|----------------------------------------------------------------------------------------------------------------------------------|--------------------------|
| <a href="#"></a><br><a href="#"><u>__hi</u></a> | Pointer to end of range. |
|----------------------------------------------------------------------------------------------------------------------------------|--------------------------|

#### Returns

Pointer to non-matching char if found, else [\\_\\_hi](#).

#### tolower() [1/2]

```
template<typename _CharT>
const char_type * std::\_\_ctype\_abstract\_base< _CharT >::tolower (
    char_type * \_\_lo,
    const char_type * \_\_hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [[\\_\\_lo](#),[\\_\\_hi](#)) to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(\_\_lo, \_\_hi)`.

#### Parameters

|                                                                                                                                  |                            |
|----------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><a href="#"><u>__lo</u></a> | Pointer to start of range. |
| <a href="#"></a><br><a href="#"><u>__hi</u></a> | Pointer to end of range.   |

#### Returns

[\\_\\_hi](#).

#### tolower() [2/2]

```
template<typename _CharT>
char_type std::\_\_ctype\_abstract\_base< _CharT >::tolower (
    char_type \_\_c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

#### Parameters

|                                                                                                                                   |                           |
|-----------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><a href="#"><u>__c</u></a> | The char_type to convert. |
|-----------------------------------------------------------------------------------------------------------------------------------|---------------------------|

#### Returns

The lowercase char\_type if convertible, else [\\_\\_c](#).

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

**toupper()** [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
    char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

**Parameters**

|                                                                                                     |                            |
|-----------------------------------------------------------------------------------------------------|----------------------------|
|  <code>__lo</code> | Pointer to start of range. |
|  <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**toupper()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
    char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper(c).

**Parameters**

|                                                                                                      |                           |
|------------------------------------------------------------------------------------------------------|---------------------------|
|  <code>__c</code> | The char_type to convert. |
|------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else `__c`.

Referenced by [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#).

**widen()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
    char __c) const [inline], [inherited]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See `codecvt` for that.



**Parameters**

|                                 |                      |
|---------------------------------|----------------------|
| <a href="#"><code>_↔</code></a> | The char to convert. |
| <code>_c</code>                 |                      |

**Returns**

The converted `char_type`.

Referenced by [std::money\\_get<\\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter >::do\\_get\(\)](#), [std::money\\_put<\\_CharT, \\_OutIter >::do\\_put\(\)](#) and [std::time\\_put<\\_CharT, \\_OutIter >::do\\_put\(\)](#).

**widen()** [2/2]

```
template<typename _CharT>
const char * std::\_\_ctype\_abstract\_base<\_CharT >::widen (
    const char * __lo,
    const char * __hi,
    char\_type * __to) const [inline], [inherited]
```

Widen array to `char_type`.

This function converts each char in the input to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                                                      |                                   |
|------------------------------------------------------|-----------------------------------|
| <a href="#"><code>_↔</code></a><br><code>__lo</code> | Pointer to start of range.        |
| <a href="#"><code>_↔</code></a><br><code>__hi</code> | Pointer to end of range.          |
| <a href="#"><code>_↔</code></a><br><code>__to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

**5.359.3 Member Data Documentation****id**

```
template<typename _CharT>
locale::id std::ctype<\_CharT >::id [static], [inherited]
```

The facet id for `ctype<char_type>`

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

**5.360 std::ctype\_byname< char > Class Reference**

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype\_byname< char >:



### Public Types

- typedef const int \* `__to_type`
- typedef char `char_type`
- typedef `ctype< char >::mask` `mask`

### Public Member Functions

- `ctype_byname` (const char \* \_\_s, size\_t \_\_refs=0)
- `ctype_byname` (const char \* \_\_s, size\_t \_\_refs=0)
- `ctype_byname` (const string & \_\_s, size\_t \_\_refs=0)
- `ctype_byname` (const string & \_\_s, size\_t \_\_refs=0)
- const char\_type \* `is` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- bool `is` (mask \_\_m, char\_type \_\_c) const
- const char\_type \* `is` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- bool `is` (mask \_\_m, char\_type \_\_c) const
- char `narrow` (char\_type \_\_c, char \_\_default) const
- const char\_type \* `narrow` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- char `narrow` (char\_type \_\_c, char \_\_default) const
- const char\_type \* `narrow` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char\_type \* `scan_is` (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* `scan_is` (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* `scan_not` (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* `scan_not` (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const

- `const mask * table () const throw ()`
- `const char_type * tolower (char_type *__lo, const char_type *__hi) const`
- `char_type tolower (char_type __c) const`
- `const char_type * tolower (char_type *__lo, const char_type *__hi) const`
- `char_type tolower (char_type __c) const`
- `const char_type * toupper (char_type *__lo, const char_type *__hi) const`
- `char_type toupper (char_type __c) const`
- `const char_type * toupper (char_type *__lo, const char_type *__hi) const`
- `char_type toupper (char_type __c) const`
- `const char * widen (const char *__lo, const char *__hi, char_type *__to) const`
- `char_type widen (char __c) const`
- `const char * widen (const char *__lo, const char *__hi, char_type *__to) const`

### Static Public Member Functions

- `static const mask * classic\_table () throw ()`

### Static Public Attributes

- `static const mask alnum`
- `static const mask alpha`
- `static const mask blank`
- `static const mask cntrl`
- `static const mask digit`
- `static const mask graph`
- `static locale::id id`
- `static const mask lower`
- `static const mask print`
- `static const mask punct`
- `static const mask space`
- `static const size_t table\_size`
- `static const mask upper`
- `static const mask xdigit`

### Protected Member Functions

- `virtual const char_type * do\_is (const char_type *__lo, const char_type *__hi, mask *__vec) const`
- `virtual bool do\_is (mask __m, char_type __c) const`
- `virtual const char_type * do\_is (const char_type *__lo, const char_type *__hi, mask *__vec) const`
- `virtual bool do\_is (mask __m, char_type __c) const`
- `virtual char do\_narrow (char_type __c, char __dfault) const`
- `virtual const char_type * do\_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const`
- `virtual char do\_narrow (char_type, char __dfault) const`
- `virtual const char_type * do\_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const`
- `virtual const char_type * do\_scan\_is (mask __m, const char_type *__lo, const char_type *__hi) const`
- `virtual const char_type * do\_scan\_is (mask __m, const char_type *__lo, const char_type *__hi) const`
- `virtual const char_type * do\_scan\_not (mask __m, const char_type *__lo, const char_type *__hi) const`
- `virtual const char_type * do\_scan\_not (mask __m, const char_type *__lo, const char_type *__hi) const`
- `virtual const char_type * do\_tolower (char_type *__lo, const char_type *__hi) const`
- `virtual char_type do\_tolower (char_type __c) const`

- virtual const char\_type \* [do\\_tolower](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_to) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_dest) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_ctype](#)
- bool [\\_M\\_del](#)
- char [\\_M\\_narrow](#) [1+static\_cast< unsigned char >(-1)]
- char [\\_M\\_narrow\\_ok](#)
- const mask \* [\\_M\\_table](#)
- \_\_to\_type [\\_M\\_tolower](#)
- \_\_to\_type [\\_M\\_toupper](#)
- char [\\_M\\_widen](#) [1+static\_cast< unsigned char >(-1)]
- char [\\_M\\_widen\\_ok](#)

#### 5.360.1 Detailed Description

22.2.1.4 Class ctype\_byname specializations.

#### 5.360.2 Member Function Documentation

##### classic\_table()

static const mask \* [std::ctype< char >::classic\\_table](#) () throw ( ) [static], [inherited]  
Returns a pointer to the C locale mask table.

##### do\_is() [1/4]

```
virtual const char_type * std::ctype< char >::do\_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [protected], [virtual], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**do\_is() [2/4]**

```
virtual bool std::ctype< char >::do_is (
    mask __m,
    char_type __c) const [protected], [virtual], [inherited]
```

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__c</code> | The <code>char_type</code> to find the mask of. |
| <code>__m</code> | The mask to compare against.                    |

**Returns**

$(M \ \& \ \_m) \neq 0$ .

**do\_is() [3/4]**

```
virtual const char_type * std::ctype< char >::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**do\_is()** [4/4]

```
virtual bool std::ctype< char >::do_is (
    mask __m,
    char_type __c) const [protected], [virtual]
```

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

**Returns**

(*M* & *\_\_m*) != 0.

**do\_narrow()** [1/4]

```
virtual char std::ctype< char >::do_narrow (
    char_type __c,
    char __dfault) const [inline], [protected], [virtual], [inherited]
```

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char to convert.                |
| <code>__dfault</code> | Char to return if conversion fails. |

**Returns**

The converted char.

**do\_narrow()** [2/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __dfault,
    char * __to) const [inline], [protected], [virtual], [inherited]
```

Narrow char array to char array.

This virtual function converts each char in the range [*lo*,*hi*) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

#### Returns

`__hi`.

#### `do_narrow()` [3/4]

```
virtual char std::ctype< char >::do_narrow (
    char_type ,
    char __dfault) const [protected], [virtual]
```

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                       |                                        |
|-----------------------|----------------------------------------|
| <code>__c</code>      | The <code>char_type</code> to convert. |
| <code>__dfault</code> | Char to return if conversion fails.    |

#### Returns

The converted `char`.

#### `do_narrow()` [4/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __dfault,
    char * __to) const [protected], [virtual]
```

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

## Returns

`__hi`.

**do\_scan\_is()** [1/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual], [inherited]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a matching char\_type if found, else `__hi`.

**do\_scan\_is()** [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |



|                                                                                                                                  |                          |
|----------------------------------------------------------------------------------------------------------------------------------|--------------------------|
| <a href="#"></a><br><a href="#"><u>__hi</u></a> | Pointer to end of range. |
|----------------------------------------------------------------------------------------------------------------------------------|--------------------------|

#### Returns

Pointer to a matching char\_type if found, else [\\_\\_hi](#).

#### do\_scan\_not() [1/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
    mask \_\_m,
    const char\_type * \_\_lo,
    const char\_type * \_\_hi) const [protected], [virtual], [inherited]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                                                                                                                                   |                              |
|-----------------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><a href="#"><u>__m</u></a>   | The mask to compare against. |
| <a href="#"></a><br><a href="#"><u>__lo</u></a>  | Pointer to start of range.   |
| <a href="#"></a><br><a href="#"><u>__hi</u></a> | Pointer to end of range.     |

#### Returns

Pointer to a non-matching char\_type if found, else [\\_\\_hi](#).

#### do\_scan\_not() [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
    mask \_\_m,
    const char\_type * \_\_lo,
    const char\_type * \_\_hi) const [protected], [virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                                                                                                                                    |                              |
|------------------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><a href="#"><u>__m</u></a>  | The mask to compare against. |
| <a href="#"></a><br><a href="#"><u>__lo</u></a> | Pointer to start of range.   |
| <a href="#"></a><br><a href="#"><u>__hi</u></a> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else \_\_hi.

**do\_tolower()** [1/4]



```
virtual const char_type * std::ctype< char >::do_tolower (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                        |                                 |
|----------------------------------------------------------------------------------------|---------------------------------|
|  __lo | Pointer to first char in range. |
|  __hi | Pointer to end of range.        |

**Returns**

\_\_hi.

References [do\\_tolower\(\)](#).

**do\_tolower()** [2/4]

```
virtual char_type std::ctype< char >::do_tolower (
    char_type __c) const [protected], [virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                         |                      |
|-----------------------------------------------------------------------------------------|----------------------|
|  __c | The char to convert. |
|-----------------------------------------------------------------------------------------|----------------------|

**Returns**

The lowercase char if convertible, else \_\_c.

References [do\\_tolower\(\)](#).

Referenced by [do\\_tolower\(\)](#), and [do\\_tolower\(\)](#).

**do\_tolower()** [3/4]

```
virtual const char_type * std::ctype< char >::do_tolower (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

#### `do_tolower()` [4/4]

```
virtual char_type std::ctype< char >::do_tolower (
    char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__c</code> | The <code>char_type</code> to convert. |
|------------------|----------------------------------------|

#### Returns

The lowercase `char_type` if convertible, else `__c`.

#### `do_toupper()` [1/4]

```
virtual const char_type * std::ctype< char >::do_toupper (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char in the range `[lo,hi)` to uppercase if possible. Other chars remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.References [do\\_toupper\(\)](#).**do\_toupper()** [2/4]

```
virtual char_type std::ctype< char >::do_toupper (
    char_type __c) const [protected], [virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

## Returns

The uppercase char if convertible, else `__c`.References [do\\_toupper\(\)](#).Referenced by [do\\_toupper\(\)](#), and [do\\_toupper\(\)](#).**do\_toupper()** [3/4]

```
virtual const char_type * std::ctype< char >::do_toupper (
    char_type * __lo,
    const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

## Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.**do\_toupper()** [4/4]

```
virtual char_type std::ctype< char >::do_toupper (
    char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__c</code> | The <code>char_type</code> to convert. |
|------------------|----------------------------------------|

#### Returns

The uppercase `char_type` if convertible, else `__c`.

#### `do_widen()` [1/3]

```
virtual const char * std::ctype< char >::do_widen (  
    const char * __lo,  
    const char * __hi,  
    char_type * __to) const [inline], [protected], [virtual], [inherited]
```

Widen char array.

This function converts each char in the range `[lo,hi)` to `char` using the simplest reasonable transformation. For an undervied `ctype<char>` facet, the argument will be copied unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

#### Returns

`__hi`.

#### `do_widen()` [2/3]

```
virtual char_type std::ctype< char >::do_widen (  
    char __c) const [protected], [virtual]
```

Widen char.

This virtual function converts the `char` to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                         |                      |
|-------------------------|----------------------|
| $\leftrightarrow$<br>_c | The char to convert. |
|-------------------------|----------------------|

## Returns

The converted char\_type

**do\_widen()** [3/3]

```
virtual const char * std::ctype< char >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __dest) const [protected], [virtual]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                           |                                   |
|---------------------------|-----------------------------------|
| $\leftrightarrow$<br>__lo | Pointer to start range.           |
| $\leftrightarrow$<br>__hi | Pointer to end of range.          |
| $\leftrightarrow$<br>__to | Pointer to the destination array. |

## Returns

\_\_hi.

**is()** [1/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

## Parameters

|       |                                      |
|-------|--------------------------------------|
| __lo  | Pointer to start of range.           |
| __hi  | Pointer to end of range.             |
| __vec | Pointer to an array of mask storage. |

## Returns

\_\_hi.

**is()** [2/4]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
    mask __m,
    char_type __c) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_↵type>::do\_is().

**Parameters**

|                          |                                       |
|--------------------------|---------------------------------------|
| <a href="#">↵</a><br>__c | The char_type to compare the mask of. |
| <a href="#">↵</a><br>__m | The mask to compare against.          |

**Returns**

(M & \_\_m) != 0.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

**is()** [3/4]

```
const char_type * std::__ctype_abstract_base< char >::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| __lo  | Pointer to start of range.           |
| __hi  | Pointer to end of range.             |
| __vec | Pointer to an array of mask storage. |

**Returns**

\_\_hi.

**is()** [4/4]

```
bool std::__ctype_abstract_base< char >::is (
    mask __m,
    char_type __c) const [inline]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_↵type>::do\_is().

## Parameters

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <a href="#"><code>__c</code></a> | The char_type to compare the mask of. |
| <a href="#"><code>__m</code></a> | The mask to compare against.          |

## Returns

(M & \_\_m) != 0.

**narrow()** [1/4]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
    char_type __c,
    char __dfault) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                                       |                                     |
|---------------------------------------|-------------------------------------|
| <a href="#"><code>__c</code></a>      | The char_type to convert.           |
| <a href="#"><code>__dfault</code></a> | Char to return if conversion fails. |

## Returns

The converted char.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), and [std::time\\_put< \\_CharT, \\_OutIter >::put\(\)](#).

**narrow()** [2/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __dfault,
    char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_dfault, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                                       |                                   |
|---------------------------------------|-----------------------------------|
| <a href="#"><code>__lo</code></a>     | Pointer to start of range.        |
| <a href="#"><code>__hi</code></a>     | Pointer to end of range.          |
| <a href="#"><code>__dfault</code></a> | Char to use if conversion fails.  |
| <a href="#"><code>__to</code></a>     | Pointer to the destination array. |



**Returns**

`__hi`.

**narrow()** [3/4]

```
char std::__ctype_abstract_base< char >::narrow (  
    char_type __c,  
    char __default) const [inline]
```

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `default` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

**Returns**

The converted `char`.

**narrow()** [4/4]

```
const char_type * std::__ctype_abstract_base< char >::narrow (  
    const char_type * __lo,  
    const char_type * __hi,  
    char __default,  
    char * __to) const [inline]
```

Narrow array to `char` array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `default` is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**scan\_is()** [1/2]

```
template<typename _CharT>  
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (  
    mask __m,
```

```
const char_type * __lo,
const char_type * __hi) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to matching char\_type if found, else `__hi`.

#### scan\_is() [2/2]

```
const char_type * std::__ctype_abstract_base< char >::scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to matching char\_type if found, else `__hi`.

#### scan\_not() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__m</code>  | The mask to compare against.    |
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**scan\_not() [2/2]**

```
const char_type * std::__ctype_abstract_base< char >::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi) const [inline]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__m</code>  | The mask to compare against.    |
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**table()**

```
const mask * std::ctype< char >::table () const throw ( ) [inline], [inherited]
```

Returns a pointer to the mask table provided to the constructor, or the default from classic\_table() if none was provided.

**tolower() [1/4]**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
    char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>:: do\_tolower(\_\_lo, \_\_hi).

## Parameters

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

**tolower()** [2/4]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
    char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

## Parameters

|                                                                                                                       |                           |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><code>__c</code> | The char_type to convert. |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|

## Returns

The lowercase char\_type if convertible, else `__c`.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

**tolower()** [3/4]

```
const char_type * std::__ctype_abstract_base< char >::tolower (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char\_type in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

## Parameters

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

**tolower()** [4/4]

```
char_type std::__ctype_abstract_base< char >::tolower (
    char_type __c) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else `__c`.

**toupper()** [1/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
    char_type * __lo,
    const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**toupper()** [2/4]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
    char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else `__c`.

Referenced by `std::time_get< _CharT, _InIter >::get()`.

**toupper()** [3/4]

```
const char_type * std::__ctype_abstract_base< char >::toupper (
    char_type * __lo,
    const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

**Parameters**

|                                                                                           |                            |
|-------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.


**toupper()** [4/4]

```
char_type std::__ctype_abstract_base< char >::toupper (
    char_type __c) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

**Parameters**

|                                                                                            |                           |
|--------------------------------------------------------------------------------------------|---------------------------|
| <br>__c | The char_type to convert. |
|--------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

**widen()** [1/3]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
    const char * __lo,
    const char * __hi,
    char_type * __to) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                             |                            |
|---------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
|---------------------------------------------------------------------------------------------|----------------------------|

**Parameters**

|                                                                                          |                                   |
|------------------------------------------------------------------------------------------|-----------------------------------|
| <br>_hi | Pointer to end of range.          |
| <br>_to | Pointer to the destination array. |

**Returns**

\_\_hi.

**widen()** [2/3]

```
char_type std::__ctype_abstract_base< char >::widen (  
    char __c) const [inline]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                         |                      |
|-----------------------------------------------------------------------------------------|----------------------|
| <br>_c | The char to convert. |
|-----------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted char\_type.

**widen()** [3/3]

```
const char * std::__ctype_abstract_base< char >::widen (  
    const char * __lo,  
    const char * __hi,  
    char_type * __to) const [inline]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                            |                                   |
|--------------------------------------------------------------------------------------------|-----------------------------------|
| <br>_lo | Pointer to start of range.        |
| <br>_hi | Pointer to end of range.          |
| <br>_to | Pointer to the destination array. |

**Returns**

\_\_hi.

### 5.360.3 Member Data Documentation

#### id

`locale::id std::ctype< char >::id` [static]

The facet id for `ctype<char_type>`

#### table\_size

`const size_t std::ctype< char >::table_size` [static], [inherited]

The size of the mask table. It is `SCHAR_MAX + 1`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.361 `__gnu_cxx::debug_allocator<_Alloc>` Class Template Reference

`#include <debug_allocator.h>`

### Public Types

- `typedef _Traits::const_pointer` **const\_pointer**
- `typedef _Traits::const_reference` **const\_reference**
- `typedef _Traits::difference_type` **difference\_type**
- `typedef _Traits::pointer` **pointer**
- `typedef _Traits::reference` **reference**
- `typedef _Traits::size_type` **size\_type**
- `typedef _Traits::value_type` **value\_type**

### Public Member Functions

- **debug\_allocator** (const `_Alloc` &\_\_a)
- `template<typename _Alloc2>`  
**debug\_allocator** (const [debug\\_allocator](#)< `_Alloc2` > &\_\_a2, `typename __convertible<_Alloc2>::__type`=0)
- `pointer` **allocate** (`size_type` \_\_n)
- `pointer` **allocate** (`size_type` \_\_n, const void \*\_\_hint)
- `template<typename _Tp, typename... _Args>`  
void **construct** (`_Tp` \*\_\_p, `_Args` &&... \_\_args)
- void **construct** (`pointer` \_\_p, const `value_type` &\_\_val)
- void **deallocate** (`pointer` \_\_p, `size_type` \_\_n)
- `template<typename _Tp>`  
void **destroy** (`_Tp` \*\_\_p)
- `size_type` **max\_size** () const throw ()

### Friends

- `template<typename _Alloc2>`  
bool **operator==** (const [debug\\_allocator](#) &\_\_lhs, const [debug\\_allocator](#)< `_Alloc2` > &\_\_rhs) noexcept



### 5.361.1 Detailed Description

```
template<typename _Alloc>
class __gnu_cxx::debug_allocator< _Alloc >
```

A meta-allocator with debugging bits.

This is precisely the allocator defined in the C++03 Standard.

The documentation for this class was generated from the following file:

- [debug\\_allocator.h](#)

## 5.362 std::decay< \_Tp > Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using **type**

### 5.362.1 Detailed Description

```
template<typename _Tp>
struct std::decay< _Tp >
```

decay

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.363 std::decimal::decimal128 Class Reference

```
#include <decimal>
```

### Public Types

- typedef float **\_\_decfloat128**

### Public Member Functions

- [decimal128](#) ([\\_\\_decfloat128](#) \_\_z)
- **decimal128** ([decimal32](#) d32)
- **decimal128** ([decimal64](#) d64)
- **decimal128** (double \_\_r)
- **decimal128** (float \_\_r)
- **decimal128** (int \_\_z)
- **decimal128** (long \_\_z)
- **decimal128** (long double \_\_r)
- **decimal128** (long long \_\_z)
- **decimal128** (unsigned int \_\_z)
- **decimal128** (unsigned long \_\_z)
- **decimal128** (unsigned long long \_\_z)
- [\\_\\_decfloat128](#) **\_\_getval** (void)
- void **\_\_setval** ([\\_\\_decfloat128](#) \_\_x)
- **operator long long** () const
- [decimal128](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator\*=** ([decimal32](#) \_\_rhs)

- [decimal128](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator\*=** (int \_\_rhs)
- [decimal128](#) & **operator\*=** (long \_\_rhs)
- [decimal128](#) & **operator\*=** (long long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator++** ()
- [decimal128](#) **operator++** (int)
- [decimal128](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator+=** (int \_\_rhs)
- [decimal128](#) & **operator+=** (long \_\_rhs)
- [decimal128](#) & **operator+=** (long long \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator--** ()
- [decimal128](#) **operator--** (int)
- [decimal128](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator-=** (int \_\_rhs)
- [decimal128](#) & **operator-=** (long \_\_rhs)
- [decimal128](#) & **operator-=** (long long \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator/=** (int \_\_rhs)
- [decimal128](#) & **operator/=** (long \_\_rhs)
- [decimal128](#) & **operator/=** (long long \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long long \_\_rhs)

### 5.363.1 Detailed Description

3.2.4 Class decimal128.

### 5.363.2 Constructor & Destructor Documentation

#### decimal128()

```
std::decimal::decimal128::decimal128 (
    __decfloat128 __z) [inline]
```

Conforming extension: Conversion from scalar decimal type.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.364 std::decimal::decimal32 Class Reference

```
#include <decimal>
```

### Public Types

- typedef float **\_\_decfloat32**

### Public Member Functions

- [decimal32](#) (\_\_decfloat32 \_\_z)
- **decimal32** ([decimal128](#) \_\_d128)
- **decimal32** ([decimal64](#) \_\_d64)
- **decimal32** (double \_\_r)
- **decimal32** (float \_\_r)
- **decimal32** (int \_\_z)
- **decimal32** (long \_\_z)
- **decimal32** (long double \_\_r)
- **decimal32** (long long \_\_z)
- **decimal32** (unsigned int \_\_z)
- **decimal32** (unsigned long \_\_z)
- **decimal32** (unsigned long long \_\_z)
- \_\_decfloat32 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat32 \_\_x)
- **operator long long** () const
- [decimal32](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator\*=** (int \_\_rhs)
- [decimal32](#) & **operator\*=** (long \_\_rhs)
- [decimal32](#) & **operator\*=** (long long \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator++** ()
- [decimal32](#) **operator++** (int)
- [decimal32](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator+=** (int \_\_rhs)
- [decimal32](#) & **operator+=** (long \_\_rhs)
- [decimal32](#) & **operator+=** (long long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator--** ()
- [decimal32](#) **operator--** (int)
- [decimal32](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator-=** (int \_\_rhs)
- [decimal32](#) & **operator-=** (long \_\_rhs)

- [decimal32](#) & **operator=** (long long \_\_rhs)
- [decimal32](#) & **operator=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator/=** (int \_\_rhs)
- [decimal32](#) & **operator/=** (long \_\_rhs)
- [decimal32](#) & **operator/=** (long long \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long long \_\_rhs)

### 5.364.1 Detailed Description

3.2.2 Class decimal32.

### 5.364.2 Constructor & Destructor Documentation

#### decimal32()

```
std::decimal::decimal32::decimal32 (
    __decfloat32 __z) [inline]
```

Conforming extension: Conversion from scalar decimal type.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.365 std::decimal::decimal64 Class Reference

```
#include <decimal>
```

### Public Types

- typedef float **\_\_decfloat64**

### Public Member Functions

- [decimal64](#) (\_\_decfloat64 \_\_z)
- [decimal64](#) ([decimal128](#) d128)
- [decimal64](#) ([decimal32](#) d32)
- [decimal64](#) (double \_\_r)
- [decimal64](#) (float \_\_r)
- [decimal64](#) (int \_\_z)
- [decimal64](#) (long \_\_z)
- [decimal64](#) (long double \_\_r)
- [decimal64](#) (long long \_\_z)
- [decimal64](#) (unsigned int \_\_z)
- [decimal64](#) (unsigned long \_\_z)
- [decimal64](#) (unsigned long long \_\_z)
- [\\_\\_decfloat64](#) **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat64 \_\_x)
- **operator long long** () const

- [decimal64](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator\*=** (int \_\_rhs)
- [decimal64](#) & **operator\*=** (long \_\_rhs)
- [decimal64](#) & **operator\*=** (long long \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator++** ()
- [decimal64](#) **operator++** (int)
- [decimal64](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator+=** (int \_\_rhs)
- [decimal64](#) & **operator+=** (long \_\_rhs)
- [decimal64](#) & **operator+=** (long long \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator--** ()
- [decimal64](#) **operator--** (int)
- [decimal64](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator-=** (int \_\_rhs)
- [decimal64](#) & **operator-=** (long \_\_rhs)
- [decimal64](#) & **operator-=** (long long \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator/=** (int \_\_rhs)
- [decimal64](#) & **operator/=** (long \_\_rhs)
- [decimal64](#) & **operator/=** (long long \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long long \_\_rhs)

### 5.365.1 Detailed Description

3.2.3 Class decimal64.

### 5.365.2 Constructor & Destructor Documentation

#### decimal64()

```
std::decimal::decimal64::decimal64 (
    __decfloat64 __z) [inline]
```

Conforming extension: Conversion from scalar decimal type.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.366 `simd_abi::deduce<_Tp, _Np,... >` Struct Template Reference

```
#include <simd.h>
```

### 5.366.1 Detailed Description

```
template<typename _Tp, size_t _Np, typename...>
struct simd_abi::deduce<_Tp, _Np,... >
```

#### Template Parameters

|                    |                                                                                                                                                                                                                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_Tp</code>   | The requested <code>value_type</code> for the elements.                                                                                                                                                                                                                                        |
| <code>_Np</code>   | The requested number of elements.                                                                                                                                                                                                                                                              |
| <code>_Abis</code> | This parameter is ignored, since this implementation cannot make any use of it. Either <code>__a</code> a good native ABI is matched and used as <code>type</code> alias, or the <code>fixed_size&lt;_Np&gt;</code> ABI is used, which internally is built from the best matching native ABIs. |

The documentation for this struct was generated from the following file:

- `simd.h`

## 5.367 `__gnu_pbds::detail::default_comb_hash_fn` Struct Reference

```
#include <standard_policies.hpp>
```

### Public Types

- typedef [direct\\_mask\\_range\\_hashing](#) type

### 5.367.1 Detailed Description

Primary template, `default_comb_hash_fn`.

### 5.367.2 Member Typedef Documentation

#### type

```
typedef direct\_mask\_range\_hashing __gnu_pbds::detail::default_comb_hash_fn::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.368 `std::default_delete<_Tp >` Struct Template Reference

```
#include <memory>
```

### Public Member Functions

- constexpr [default\\_delete](#) () noexcept=default
- template<typename \_Up, typename = \_Require<is\_convertible<\_Up\*, \_Tp\*>>>  
constexpr [default\\_delete](#) (const [default\\_delete](#)<\_Up > &) noexcept
- constexpr void [operator\(\)](#) (\_Tp \*\_\_ptr) const

### 5.368.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete<_Tp>
```

Primary template of `default_delete`, used by `unique_ptr` for single objects

Since

C++11

### 5.368.2 Constructor & Destructor Documentation

#### `default_delete()` [1/2]

```
template<typename _Tp>
std::default_delete<_Tp>::default_delete() [constexpr], [default], [noexcept]
Default constructor.
Referenced by default\_delete\(\).
```

#### `default_delete()` [2/2]

```
template<typename _Tp>
template<typename _Up, typename = _Require<is_convertible<_Up*, _Tp*>>>
std::default_delete<_Tp>::default_delete (
    const default_delete<_Up> & ) [inline], [constexpr], [noexcept]
Converting constructor.
Allows conversion from a deleter for objects of another type, _Up, only if _Up* is convertible to _Tp*.
References default\_delete\(\).
```

### 5.368.3 Member Function Documentation

#### `operator>()`

```
template<typename _Tp>
void std::default_delete<_Tp>::operator() (
    _Tp * __ptr) const [inline], [constexpr]
Calls delete __ptr
The documentation for this struct was generated from the following file:
```

- [unique\\_ptr.h](#)

## 5.369 `std::default_delete<_Tp[]>` Struct Template Reference

```
#include <memory>
```

### Public Member Functions

- constexpr [default\\_delete](#) () noexcept=default
- constexpr [default\\_delete](#) () noexcept=default
- constexpr [default\\_delete](#) (const [default\\_delete](#)<\_Up> &) noexcept
- template<typename \_Up, typename = \_Require<is\_convertible<\_Up(\*)[], \_Tp(\*)[]>>> constexpr [default\\_delete](#) (const [default\\_delete](#)<\_Up[]> &) noexcept
- constexpr void [operator\(\)](#) (\_Tp \*\_\_ptr) const
- template<typename \_Up> constexpr [enable\\_if](#)< is\_convertible<\_Up(\*)[], \_Tp(\*)[]>::value >::type [operator\(\)](#) (\_Up \*\_\_ptr) const

### 5.369.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete<_Tp[]>
```

Specialization of default\_delete for arrays, used by unique\_ptr<T[]>

Since

C++11

### 5.369.2 Constructor & Destructor Documentation

#### default\_delete() [1/4]

```
template<typename _Tp>
std::default_delete<_Tp[]>::default_delete () [constexpr], [default], [noexcept]
Default constructor.
Referenced by default\_delete\(\).
```

#### default\_delete() [2/4]

```
template<typename _Tp>
template<typename _Up, typename = _Require<is_convertible<_Up(*)[], _Tp(*)[]>>>
std::default_delete<_Tp[]>::default_delete (
    const default_delete<_Up[]> & ) [inline], [constexpr], [noexcept]
```

Converting constructor.

Allows conversion from a deleter for arrays of another type, such as a const-qualified version of \_Tp.

Conversions from types derived from \_Tp are not allowed because it is undefined to delete[] an array of derived types through a pointer to the base type.

References [default\\_delete\(\)](#).

#### default\_delete() [3/4]

```
std::default_delete<_Tp >::default_delete () [constexpr], [default], [noexcept]
Default constructor.
```

#### default\_delete() [4/4]

```
std::default_delete<_Tp >::default_delete (
    const default_delete<_Up > & ) [inline], [constexpr], [noexcept]
```

Converting constructor.

Allows conversion from a deleter for objects of another type, \_Up, only if \_Up\* is convertible to \_Tp\*.

### 5.369.3 Member Function Documentation

#### operator>() [1/2]

```
void std::default_delete<_Tp >::operator() (
    _Tp * __ptr) const [inline], [constexpr]
Calls delete __ptr
```

#### operator>() [2/2]

```
template<typename _Tp>
template<typename _Up>
```



```
enable_if< is_convertible< _Up(*)[], _Tp(*)[]>::value >::type std::default_delete< _Tp[]>↵  
::operator() (↵  
    _Up * __ptr) const [inline], [constexpr]
```

Calls delete[] \_\_ptr

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 5.370 \_\_gnu\_pbds::detail::default\_eq\_fn< Key > Struct Template Reference

```
#include <standard_policies.hpp>
```

### Public Types

- typedef [std::equal\\_to](#)< Key > [type](#)

#### 5.370.1 Detailed Description

```
template<typename Key>
```

```
struct __gnu_pbds::detail::default_eq_fn< Key >
```

Primary template, default\_eq\_fn.

#### 5.370.2 Member Typedef Documentation

##### type

```
template<typename Key>
```

```
typedef std::equal\_to<Key> \_\_gnu\_pbds::detail::default\_eq\_fn< Key >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.371 \_\_gnu\_pbds::detail::default\_hash\_fn< Key > Struct Template Reference

```
#include <standard_policies.hpp>
```

### Public Types

- typedef [std::tr1::hash](#)< Key > [type](#)

#### 5.371.1 Detailed Description

```
template<typename Key>
```

```
struct __gnu_pbds::detail::default_hash_fn< Key >
```

Primary template, default\_hash\_fn.

#### 5.371.2 Member Typedef Documentation

##### type

```
template<typename Key>
```

```
typedef std::tr1::hash<Key> \_\_gnu\_pbds::detail::default\_hash\_fn< Key >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.372 `__gnu_parallel::default_parallel_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



### Public Member Functions

- `default_parallel_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

### 5.372.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

### 5.372.2 Member Function Documentation

#### `__get_num_threads()`

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

#### `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
    _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

#### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.373 `__gnu_pbds::detail::default_probe_fn`< `Comb_Probe_Fn` > Struct Template Reference

```
#include <standard_policies.hpp>
```

#### Public Types

- typedef `cond_type::__type` [type](#)

#### 5.373.1 Detailed Description

```
template<typename Comb_Probe_Fn>
struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >
```

Primary template, `default_probe_fn`.

#### 5.373.2 Member Typedef Documentation

##### type

```
template<typename Comb_Probe_Fn>
typedef cond_type::__type \_\_gnu\_pbds::detail::default\_probe\_fn< Comb_Probe_Fn >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

### 5.374 `__gnu_pbds::detail::default_resize_policy`< `Comb_Hash_Fn` > Struct Template Reference

```
#include <standard_policies.hpp>
```

#### Public Types

- typedef [hash\\_standard\\_resize\\_policy](#)< `size_policy_type`, [trigger](#), `false`, `size_type` > [type](#)

#### 5.374.1 Detailed Description

```
template<typename Comb_Hash_Fn>
struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >
```

Primary template, `default_resize_policy`.

#### 5.374.2 Member Typedef Documentation

##### type

```
template<typename Comb_Hash_Fn>
typedef hash\_standard\_resize\_policy<size_policy_type, trigger, false, size_type> \_\_gnu\_pbds::detail::default\_res
Comb_Hash_Fn >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

### 5.375 `std::default_sentinel_t` Struct Reference

```
#include <iterator_concepts.h>
```

### 5.375.1 Detailed Description

A sentinel type that can be used to check for the end of a range.

For some iterator types the past-the-end sentinel value is independent of the underlying sequence, and a default sentinel can be used with them. For example, a `std::counted_iterator` keeps a count of how many elements remain, and so checking for the past-the-end value only requires checking if that count has reached zero. A past-the-end `std::istream_iterator` is equal to the default-constructed value, which can be easily checked.

Comparing iterators of these types to `std::default_sentinel` is a convenient way to check if the end has been reached.

Since

C++20

The documentation for this struct was generated from the following file:

- [iterator\\_concepts.h](#)

## 5.376 `__gnu_pbds::detail::default_trie_access_traits< Key >` Struct Template Reference

### 5.376.1 Detailed Description

`template<typename Key>`

`struct __gnu_pbds::detail::default_trie_access_traits< Key >`

Primary template, `default_trie_access_traits`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.377 `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >` Struct Template Reference

```
#include <standard_policies.hpp>
```

### Public Types

- typedef `trie_string_access_traits< string_type > type`

### 5.377.1 Detailed Description

`template<typename Char, typename Char_Traits>`

`struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >`

Partial specialization, `default_trie_access_traits`.

### 5.377.2 Member Typedef Documentation

#### type

`template<typename Char, typename Char_Traits>`

typedef `trie_string_access_traits<string_type> __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >::type`

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.378 `__gnu_pbds::detail::default_update_policy` Struct Reference

```
#include <standard_policies.hpp>
```

### Public Types

- typedef [lu\\_move\\_to\\_front\\_policy](#) type

#### 5.378.1 Detailed Description

Default update policy.

#### 5.378.2 Member Typedef Documentation

##### type

```
typedef lu\_move\_to\_front\_policy __gnu_pbds::detail::default_update_policy::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.379 `std::defer_lock_t` Struct Reference

```
#include <std_mutex.h>
```

#### 5.379.1 Detailed Description

Do not acquire ownership of the mutex.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

## 5.380 `std::__debug::deque<_Tp, _Allocator>` Class Template Reference

```
#include <deque>
```

Inheritance diagram for `std::__debug::deque<_Tp, _Allocator>`:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< [\\_Base\\_const\\_iterator](#), [deque](#) > **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**

- typedef [std::reverse\\_iterator](#)< [const\\_iterator](#) > **const\_reverse\_iterator**
- typedef [\\_Base::difference\\_type](#) **difference\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< [\\_Base\\_iterator](#), [deque](#) > **iterator**
- typedef [\\_Base::pointer](#) **pointer**
- typedef [\\_Base::reference](#) **reference**
- typedef [std::reverse\\_iterator](#)< [iterator](#) > **reverse\_iterator**
- typedef [\\_Base::size\\_type](#) **size\_type**
- typedef [\\_Tp](#) **value\_type**

## Public Member Functions

- **deque** ([\\_Base\\_ref](#) \_\_x)
- template<class [\\_InputIterator](#), typename = [std::\\_RequireInputIter](#)<[\\_InputIterator](#)>>  
**deque** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **deque** (const [\\_Allocator](#) &\_\_a)
- **deque** (const [deque](#) &)=default
- **deque** (const [deque](#) &\_\_d, const [\\_\\_type\\_identity\\_t](#)< [\\_Allocator](#) > &\_\_a)
- **deque** ([deque](#) &&)=default
- **deque** ([deque](#) &&\_\_d, const [\\_\\_type\\_identity\\_t](#)< [\\_Allocator](#) > &\_\_a)
- **deque** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **deque** ([size\\_type](#) \_\_n, const [\\_\\_type\\_identity\\_t](#)< [\\_Tp](#) > &\_\_value, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **deque** ([size\\_type](#) \_\_n, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- const [\\_Base](#) & **\_M\_base** () const noexcept
- [\\_Base](#) & **\_M\_base** () noexcept
- template<class [\\_InputIterator](#), typename = [std::\\_RequireInputIter](#)<[\\_InputIterator](#)>>  
void **assign** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- void **assign** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- void **assign** ([size\\_type](#) \_\_n, const [\\_Tp](#) &\_\_t)
- const [reference](#) **back** () const noexcept
- [reference](#) **back** () noexcept
- const [iterator](#) **begin** () const noexcept
- [iterator](#) **begin** () noexcept
- const [iterator](#) **cbegin** () const noexcept
- const [iterator](#) **cend** () const noexcept
- void **clear** () noexcept
- const [reverse\\_iterator](#) **crbegin** () const noexcept
- const [reverse\\_iterator](#) **crend** () const noexcept
- template<typename... [\\_Args](#)>  
[iterator](#) **emplace** ([const\\_iterator](#) \_\_position, [\\_Args](#) &&... \_\_args)
- template<typename... [\\_Args](#)>  
[reference](#) **emplace\_back** ([\\_Args](#) &&... \_\_args)
- template<typename... [\\_Args](#)>  
[reference](#) **emplace\_front** ([\\_Args](#) &&... \_\_args)
- const [iterator](#) **end** () const noexcept
- [iterator](#) **end** () noexcept
- [iterator](#) **erase** ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
- [iterator](#) **erase** ([const\\_iterator](#) \_\_position)
- const [reference](#) **front** () const noexcept
- [reference](#) **front** () noexcept
- template<class [\\_InputIterator](#), typename = [std::\\_RequireInputIter](#)<[\\_InputIterator](#)>>  
[iterator](#) **insert** ([const\\_iterator](#) \_\_position, [\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)

- [iterator insert](#) ([const\\_iterator](#) \_\_position, [\\_Tp](#) &&\_\_x)
- [iterator insert](#) ([const\\_iterator](#) \_\_position, [const \\_Tp](#) &\_\_x)
- [iterator insert](#) ([const\\_iterator](#) \_\_position, [initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [iterator insert](#) ([const\\_iterator](#) \_\_position, [size\\_type](#) \_\_n, [const \\_Tp](#) &\_\_x)
- [deque & operator=](#) ([const deque](#) &)=default
- [deque & operator=](#) ([deque](#) &&)=default
- [deque & operator=](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [const\\_reference operator\[\]](#) ([size\\_type](#) \_\_n) [const](#) [noexcept](#)
- [reference operator\[\]](#) ([size\\_type](#) \_\_n) [noexcept](#)
- [void pop\\_back](#) () [noexcept](#)
- [void pop\\_front](#) () [noexcept](#)
- [void push\\_back](#) ([\\_Tp](#) &&\_\_x)
- [void push\\_back](#) ([const \\_Tp](#) &\_\_x)
- [void push\\_front](#) ([\\_Tp](#) &&\_\_x)
- [void push\\_front](#) ([const \\_Tp](#) &\_\_x)
- [const\\_reverse\\_iterator rbegin](#) () [const](#) [noexcept](#)
- [reverse\\_iterator rbegin](#) () [noexcept](#)
- [const\\_reverse\\_iterator rend](#) () [const](#) [noexcept](#)
- [reverse\\_iterator rend](#) () [noexcept](#)
- [void resize](#) ([size\\_type](#) \_\_sz)
- [void resize](#) ([size\\_type](#) \_\_sz, [const \\_Tp](#) &\_\_c)
- [void shrink\\_to\\_fit](#) () [noexcept](#)
- [void swap](#) ([deque](#) &\_\_x) [noexcept](#)(/\*[conditional](#) \*/)

### Protected Member Functions

- [constexpr void \\_M\\_swap](#) ([const \\_Safe\\_container](#) &\_\_x) [const](#) [noexcept](#)

### Friends

- [template](#)<[typename](#) \_\_ItT, [typename](#) \_\_SeqT, [typename](#) \_\_CatT>  
class ::[\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)

#### 5.380.1 Detailed Description

[template](#)<[typename](#) \_\_Tp, [typename](#) \_\_Allocator = [std::allocator](#)<\_\_Tp>>  
[class](#) [std::\\_\\_debug::deque](#)< \_\_Tp, \_\_Allocator >

Class [std::deque](#) with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [debug/deque](#)

#### 5.381 [std::deque](#)< \_\_Tp, \_\_Alloc > Class Template Reference

```
#include <stl_deque.h>
```

Inheritance diagram for std::deque<\_Tp, \_Alloc>:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `deque()`=default
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`deque(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `deque(const allocator_type &__a)`
- `deque(const deque &__x)`
- `deque(const deque &__x, const __type_identity_t< allocator_type > &__a)`
- `deque(deque &&)=default`
- `deque(deque &&__x, const __type_identity_t< allocator_type > &__a)`
- `deque(initializer_list< value_type > __l, const allocator_type &__a=allocator_type())`
- `deque(size_type __n, const allocator_type &__a=allocator_type())`
- `deque(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `~deque()`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`void assign(_InputIterator __first, _InputIterator __last)`
- `void assign(initializer_list< value_type > __l)`
- `void assign(size_type __n, const value_type &__val)`
- `reference at(size_type __n)`



- const\_reference [at](#) (size\_type \_\_n) const
- const\_reference [back](#) () const noexcept
- reference [back](#) () noexcept
- [const\\_iterator begin](#) () const noexcept
- [iterator begin](#) () noexcept
- [const\\_iterator cbegin](#) () const noexcept
- [const\\_iterator cend](#) () const noexcept
- void [clear](#) () noexcept
- [const\\_reverse\\_iterator crbegin](#) () const noexcept
- [const\\_reverse\\_iterator crend](#) () const noexcept
- template<typename... \_Args>  
[iterator emplace](#) (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
reference **emplace\_back** (\_Args &&... \_\_args)
- template<typename... \_Args>  
reference **emplace\_front** (\_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- [const\\_iterator end](#) () const noexcept
- [iterator end](#) () noexcept
- [iterator erase](#) (const\_iterator \_\_first, const\_iterator \_\_last)
- [iterator erase](#) (const\_iterator \_\_position)
- const\_reference [front](#) () const noexcept
- reference [front](#) () noexcept
- allocator\_type [get\\_allocator](#) () const noexcept
- [iterator insert](#) (const\_iterator \_\_p, initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>  
[iterator insert](#) (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator insert](#) (const\_iterator \_\_position, const value\_type &\_\_x)
- [iterator insert](#) (const\_iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- [iterator insert](#) (const\_iterator \_\_position, value\_type &&\_\_x)
- size\_type [max\\_size](#) () const noexcept
- deque & operator= (const deque &\_\_x)
- deque & operator= (deque &&\_\_x) noexcept(\_Alloc\_traits::\_S\_always\_equal())
- deque & operator= (initializer\_list< value\_type > \_\_l)
- const\_reference operator[] (size\_type \_\_n) const noexcept
- reference operator[] (size\_type \_\_n) noexcept
- void [pop\\_back](#) () noexcept
- void [pop\\_front](#) () noexcept
- void [push\\_back](#) (const value\_type &\_\_x)
- void **push\_back** (value\_type &&\_\_x)
- void [push\\_front](#) (const value\_type &\_\_x)
- void **push\_front** (value\_type &&\_\_x)
- [const\\_reverse\\_iterator rbegin](#) () const noexcept
- [reverse\\_iterator rbegin](#) () noexcept
- [const\\_reverse\\_iterator rend](#) () const noexcept
- [reverse\\_iterator rend](#) () noexcept
- void [resize](#) (size\_type \_\_new\_size)
- void [resize](#) (size\_type \_\_new\_size, const value\_type &\_\_x)
- void [shrink\\_to\\_fit](#) () noexcept
- size\_type [size](#) () const noexcept
- void [swap](#) (deque &\_\_x) noexcept

**Protected Types**

- enum { `_S_initial_map_size` }
- typedef `__gnu_cxx::__alloc_traits<_Map_alloc_type> _Map_alloc_traits`
- typedef `_Alloc_traits::template rebind<_Ptr>::other _Map_alloc_type`
- typedef `_Alloc_traits::pointer _Ptr`
- typedef `_Alloc_traits::const_pointer _Ptr_const`

**Protected Member Functions**

- template<typename `_ForwardIterator`>  
void `_M_assign_aux` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `std::forward_iterator_tag`)
- template<typename `_InputIterator`>  
void `_M_assign_aux` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `std::input_iterator_tag`)
- void `_M_deallocate_map` (`_Map_pointer` \_\_p, `size_t` \_\_n) noexcept
- void `_M_default_append` (`size_type` \_\_n)
- void `_M_default_initialize` ()
- template<typename `_Alloc1`>  
void `_M_destroy_data` (`iterator` \_\_first, `iterator` \_\_last, const `_Alloc1` &)
- void `_M_destroy_data` (`iterator` \_\_first, `iterator` \_\_last, const `std::allocator<_Tp>` &)
- void `_M_destroy_data_aux` (`iterator` \_\_first, `iterator` \_\_last)
- template<typename... `_Args`>  
`iterator` `_M_emplace_aux` (`iterator` \_\_pos, `_Args` &&... \_\_args)
- `iterator` `_M_erase` (`iterator` \_\_first, `iterator` \_\_last)
- `iterator` `_M_erase` (`iterator` \_\_pos)
- void `_M_erase_at_begin` (`iterator` \_\_pos)
- void `_M_erase_at_end` (`iterator` \_\_pos)
- void `_M_fill_assign` (`size_type` \_\_n, const `value_type` &\_\_val)
- void `_M_fill_initialize` (const `value_type` &\_\_value)
- void `_M_fill_insert` (`iterator` \_\_pos, `size_type` \_\_n, const `value_type` &\_\_x)
- `_Map_alloc_type` `_M_get_map_allocator` () const noexcept
- template<typename `_ForwardIterator`>  
void `_M_insert_aux` (`iterator` \_\_pos, `_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `size_type` \_\_n)
- `iterator` `_M_insert_aux` (`iterator` \_\_pos, const `value_type` &\_\_x)
- void `_M_insert_aux` (`iterator` \_\_pos, `size_type` \_\_n, const `value_type` &\_\_x)
- void `_M_move_assign1` (`deque` &&\_\_x, `false_type`)
- void `_M_move_assign1` (`deque` &&\_\_x, `true_type`) noexcept
- void `_M_move_assign2` (`deque` &&\_\_x, `false_type`)
- void `_M_move_assign2` (`deque` &&\_\_x, `true_type`)
- template<typename `_InputIterator`, typename `_Sentinel`>  
void `_M_range_append` (`_InputIterator` \_\_first, `_Sentinel` \_\_last, `size_type` \_\_n)
- void `_M_range_check` (`size_type` \_\_n) const
- template<typename `_ForwardIterator`>  
void `_M_range_insert_aux` (`iterator` \_\_pos, `_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `std::forward_iterator_tag`)
- template<typename `_InputIterator`>  
void `_M_range_insert_aux` (`iterator` \_\_pos, `_InputIterator` \_\_first, `_InputIterator` \_\_last, `std::input_iterator_tag`)
- template<typename `_InputIterator`, typename `_Sentinel`>  
void `_M_range_prepend` (`_InputIterator` \_\_first, `_Sentinel` \_\_last, `size_type` \_\_n)
- template<typename... `_Args`>  
void `_M_replace_map` (`_Args` &&... \_\_args)
- bool `_M_shrink_to_fit` ()

- `template<typename _InputIterator>`  
`void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator>`  
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename... _Args>`  
`void _M_push_back_aux (_Args &&... __args)`
- `template<typename... _Args>`  
`void _M_push_front_aux (_Args &&... __args)`
- `void _M_pop_back_aux ()`
- `void _M_pop_front_aux ()`
- `iterator _M_reserve_elements_at_front (size_type __n)`
- `iterator _M_reserve_elements_at_back (size_type __n)`
- `void _M_new_elements_at_front (size_type __new_elements)`
- `void _M_new_elements_at_back (size_type __new_elements)`
- `void _M_reserve_map_at_back (size_type __nodes_to_add=1)`
- `void _M_reserve_map_at_front (size_type __nodes_to_add=1)`
- `void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)`

### Static Protected Member Functions

- `static size_t _S_check_init_len (size_t __n, const allocator_type &__a)`
- `static size_type _S_max_size (const _Tp_alloc_type &__a) noexcept`

#### 5.381.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`class std::deque<_Tp, _Alloc >`

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a `deque<Tp>` manages memory. Each deque has 4 members:

- `Tp** _M_map`
- `size_t _M_map_size`
- `iterator _M_start, _M_finish`

map\_size is at least 8. map is an array of map\_size pointers-to-nodes. (The name map has nothing to do with the std::map class, and **nodes** should not be confused with std::list's usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-Tp. If Tp is very large, there will be one Tp element per node (i.e., an *array* of one). For non-huge Tp's, node size is inversely related to Tp size: the larger the Tp, the fewer Tp's will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different Tp's, to improve allocator efficiency.

Not every pointer in the map array will point to a node. If the initial number of elements in the deque is small, the /middle/ map pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the map grows: available map pointers, if any, will be on the ends. As new nodes are created, only a subset of the map's pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator i:
  - i.node points to a member of the map array. (Yes, you read that correctly: i.node does not actually point to a node.) The member of the map array is what actually points to the node.
  - i.first == \*(i.node) (This points to the node (first Tp element).)
  - i.last == i.first + node\_size
  - i.cur is a pointer in the range [i.first, i.last). NOTE: the implication of this is that i.cur is always a dereferenceable pointer, even if i is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.↔node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.↔first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map\_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map\_size).
- A pointer in the range [map, map + map\_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, \_Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

### 5.381.2 Constructor & Destructor Documentation

#### deque() [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque () [default]
```

Creates a deque with no elements.

Referenced by [insert\(\)](#), and [operator=\(\)](#).

#### deque() [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    const allocator_type & __a) [inline], [explicit]
```

Creates a deque with no elements.

## Parameters

|                                   |                      |
|-----------------------------------|----------------------|
| <code>↔</code><br><code>_a</code> | An allocator object. |
|-----------------------------------|----------------------|

**deque()** [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    size_type __n,
    const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a deque with default constructed elements.

## Parameters

|                                   |                                             |
|-----------------------------------|---------------------------------------------|
| <code>↔</code><br><code>_n</code> | The number of elements to initially create. |
| <code>↔</code><br><code>_a</code> | An allocator.                               |

This constructor fills the deque with *n* default constructed elements.

**deque()** [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    size_type __n,
    const value_type & __value,
    const allocator_type & __a = allocator_type()) [inline]
```

Creates a deque with copies of an exemplar element.

## Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator.                               |

This constructor fills the deque with `__n` copies of `__value`.

**deque()** [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    const deque< _Tp, _Alloc > & __x) [inline]
```

Deque copy constructor.

## Parameters

|                                   |                                                   |
|-----------------------------------|---------------------------------------------------|
| <code>↔</code><br><code>_x</code> | A deque of identical element and allocator types. |
|-----------------------------------|---------------------------------------------------|

The newly-created deque uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

**deque()** [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc >::deque (
    deque<_Tp, _Alloc > && ) [default]
```

Deque move constructor.

The newly-created deque contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified deque.

**deque()** [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc >::deque (
    const deque<_Tp, _Alloc > & __x,
    const __type_identity_t< allocator_type > & __a) [inline]
```

Copy constructor with alternative allocator.

**deque()** [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc >::deque (
    deque<_Tp, _Alloc > && __x,
    const __type_identity_t< allocator_type > & __a) [inline]
```

Move constructor with alternative allocator.

**deque()** [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc >::deque (
    initializer_list< value_type > __l,
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a deque from an initializer list.

**Parameters**

|                          |                      |
|--------------------------|----------------------|
| $\leftrightarrow$<br>__l | An initializer_list. |
| $\leftrightarrow$<br>__a | An allocator object. |

Create a deque consisting of copies of the elements in the initializer\_list \_\_l.

This will call the element type's copy constructor N times (where N is \_\_l.size()) and do no memory reallocation.

**deque()** [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::deque<_Tp, _Alloc >::deque (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a deque from a range.

**Parameters**

|         |                    |
|---------|--------------------|
| __first | An input iterator. |
|---------|--------------------|

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__last</code> | An input iterator.   |
| <code>__a</code>    | An allocator object. |

Create a deque consisting of copies of the elements from `[__first, __last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor *N* times (where *N* is `distance(__first, __last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most *2N* calls to the copy constructor, and  $\log N$  memory reallocations.

**`~deque()`**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::~~deque () [inline]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**5.381.3 Member Function Documentation****`_M_fill_initialize()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_fill_initialize (
    const value_type & __value) [protected]
```

Fills the deque with copies of `value`.

**Parameters**

|                      |                |
|----------------------|----------------|
| <code>__value</code> | Initial value. |
|----------------------|----------------|

**Returns**

Nothing.

**Precondition**

`_M_start` and `_M_finish` have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

References [std::\\_Destroy\(\)](#).

Referenced by [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::deque\(\)](#).

**`_M_new_elements_at_back()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_new_elements_at_back (
    size_type __new_elements) [protected]
```

Memory-handling helpers for the previous internal insert functions.

References [\\_M\\_reserve\\_map\\_at\\_back\(\)](#), [max\\_size\(\)](#), and [std::size\(\)](#).

Referenced by [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::\\_M\\_reserve\\_elements\\_at\\_back\(\)](#).

**`_M_new_elements_at_front()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_new_elements_at_front (
    size_type __new_elements) [protected]
```

Memory-handling helpers for the previous internal insert functions.

References [max\\_size\(\)](#), and [std::size\(\)](#).

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::\\_M\\_reserve\\_elements\\_at\\_front\(\)](#).

**`_M_pop_back_aux()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_pop_back_aux () [protected]
```

Helper functions for `push_*` and `pop_*`.

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::pop\\_back\(\)](#).

**`_M_pop_front_aux()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_pop_front_aux () [protected]
```

Helper functions for `push_*` and `pop_*`.

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::pop\\_front\(\)](#).

**`_M_push_back_aux()`**

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
void deque::_M_push_back_aux (
    _Args &&... __args) [protected]
```

Helper functions for `push_*` and `pop_*`.

References [\\_M\\_reserve\\_map\\_at\\_back\(\)](#), [std::forward\(\)](#), [max\\_size\(\)](#), and [std::size\(\)](#).

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::push\\_back\(\)](#).

**`_M_push_front_aux()`**

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
void deque::_M_push_front_aux (
    _Args &&... __args) [protected]
```

Helper functions for `push_*` and `pop_*`.

References [\\_M\\_reserve\\_map\\_at\\_front\(\)](#), [std::forward\(\)](#), [max\\_size\(\)](#), and [std::size\(\)](#).

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::push\\_front\(\)](#).

**`_M_range_check()`**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::_M_range_check (
    size_type __n) const [inline], [protected]
```

Safety check used only from `at()`.

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::at\(\)](#), and [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::at\(\)](#).

**`_M_range_initialize()` [1/2]**

```
template<typename _Tp, typename _Alloc>
template<typename _ForwardIterator>
```



```
void deque::_M_range_initialize (
    _ForwardIterator __first,
    _ForwardIterator __last,
    std::forward_iterator_tag ) [protected]
```

Fills the deque with whatever is in [first,last).

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

#### Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

References [std::\\_Destroy\(\)](#), [std::advance\(\)](#), and [std::distance\(\)](#).

#### **`_M_range_initialize()`** [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator>
void deque::_M_range_initialize (
    _InputIterator __first,
    _InputIterator __last,
    std::input_iterator_tag ) [protected]
```

Fills the deque with whatever is in [first,last).

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

#### Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

References [clear\(\)](#), and [push\\_back\(\)](#).

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::deque\(\)](#), and [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::de](#)

#### **`_M_reallocate_map()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_reallocate_map (
    size_type __nodes_to_add,
    bool __add_at_front) [protected]
```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers.

(And consequently, deque iterators.)

References [std::max\(\)](#), and [max\\_size\(\)](#).

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::\\_M\\_reserve\\_map\\_at\\_back\(\)](#), and [std::deque<\\_Tp, polymorphic\\_all](#)

**\_M\_reserve\_elements\_at\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::_M_reserve_elements_at_back (
    size_type __n) [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

**\_M\_reserve\_elements\_at\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::_M_reserve_elements_at_front (
    size_type __n) [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

**\_M\_reserve\_map\_at\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::_M_reserve_map_at_back (
    size_type __nodes_to_add = 1) [inline], [protected]
```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Referenced by `_M_new_elements_at_back()`, and `_M_push_back_aux()`.

**\_M\_reserve\_map\_at\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::_M_reserve_map_at_front (
    size_type __nodes_to_add = 1) [inline], [protected]
```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Referenced by `_M_push_front_aux()`.

**assign() [1/3]**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::deque<_Tp, _Alloc>::assign (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Assigns a range to a deque.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a deque with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

**assign()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::assign (
    initializer_list< value_type > __l) [inline]
```

Assigns an initializer list to a deque.

**Parameters**

|   |                      |
|---|----------------------|
| ↔ | An initializer_list. |
| ↔ |                      |
| ↔ |                      |
| ↔ |                      |
| / |                      |

This function fills a deque with copies of the elements in the initializer\_list \_\_l.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

**assign()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::assign (
    size_type __n,
    const value_type & __val) [inline]
```

Assigns a given value to a deque.

**Parameters**

|       |                                    |
|-------|------------------------------------|
| __n   | Number of elements to be assigned. |
| __val | Value to be assigned.              |

This function fills a deque with *n* copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

**at()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque<_Tp, _Alloc>::at (
    size_type __n) [inline]
```

Provides access to the data contained in the deque.

**Parameters**

|     |                                                             |
|-----|-------------------------------------------------------------|
| ↔   | The index of the element for which data should be accessed. |
| __n |                                                             |

**Returns**

Read/write reference to data.

**Exceptions**

|                          |                             |
|--------------------------|-----------------------------|
| <i>std::out_of_range</i> | If __n is an invalid index. |
|--------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

### at() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::at (
    size_type __n) const [inline]
```

Provides access to the data contained in the deque.

#### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

#### Returns

Read-only (constant) reference to data.

#### Exceptions

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

### back() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::back () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the deque.

### back() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::back () [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the last element of the deque.

### begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::begin () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

### begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::begin () [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Referenced by `std::deque< _Tp, polymorphic_allocator< _Tp > >::deque()`, `std::deque< _Tp, polymorphic_allocator< _Tp > >::deque`  
`std::deque< _Tp, polymorphic_allocator< _Tp > >::~deque()`, `std::deque< _Tp, polymorphic_allocator< _Tp > >::clear()`,

`std::deque< _Tp, polymorphic_allocator< _Tp > >::front()`, `std::deque< _Tp, polymorphic_allocator< _Tp > >::front()`, `std::deque< _Tp, polymorphic_allocator< _Tp > >::insert()`, `std::deque< _Tp, polymorphic_allocator< _Tp > >::insert()`, `std::deque< _Tp, polymorphic_allocator< _Tp > >::insert()`, `std::operator<=>()`, and `operator=()`.

### **cbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Referenced by `std::deque< _Tp, polymorphic_allocator< _Tp > >::insert()`, `std::deque< _Tp, polymorphic_allocator< _Tp > >::insert()` and `std::deque< _Tp, polymorphic_allocator< _Tp > >::insert()`.

### **cend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

### **clear()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::clear () [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by `_M_range_initialize()`.

### **crbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

### **crend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

### **emplace()**

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
deque< _Tp, _Alloc >::iterator deque::emplace (
    const_iterator __position,
    _Args &&... __args)
```

Inserts an object in deque before specified iterator.

#### **Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the deque. |
| <code>__args</code>     | Arguments.                                    |

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type T constructed with `T(std::forward<Args>(args)...) before the specified location.`

References [std::forward\(\)](#).

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\(\)](#).

**empty()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::deque<_Tp, _Alloc>::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the deque is empty. (Thus `begin()` would equal `end()`.)

**end()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque<_Tp, _Alloc>::end () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

**end()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::end () [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::deque\(\)](#), [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::deque\(const\\_iterator, const\\_iterator\)](#), [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::~deque\(\)](#), [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::back\(\)](#), [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::back\(\)](#), [std::operator<=>\(\)](#), and [operator=\(\)](#).

**erase()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::erase (
    const_iterator __first,
    const_iterator __last) [inline]
```

Remove a range of elements.

**Parameters**

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

**Returns**

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[__first, __last)` and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::erase (
    const_iterator __position) [inline]
```

Remove element at given position.

**Parameters**

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

**Returns**

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**front()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::front () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the deque.

**front()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::front () [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the first element of the deque.

**get\_allocator()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::deque< _Tp, _Alloc >::get_allocator () const [inline], [nodiscard], [noexcept]
```

Get a copy of the memory allocation object.

Referenced by [operator=\(\)](#).

**insert()** [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
    const_iterator __p,
    initializer_list< value_type > __l) [inline]
```

Inserts an initializer list into the deque.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | An iterator into the deque. |
| <code>__l</code> | An initializer_list.        |

**Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the initializer\_list `__l` into the deque before the location specified by `__p`. This is known as *list insert*.

**insert()** [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::deque< _Tp, _Alloc >::insert (
    const_iterator __position,
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Inserts a range into the deque.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__position</code> | A const_iterator into the deque. |
| <code>__first</code>    | An input iterator.               |
| <code>__last</code>     | An input iterator.               |

**Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the range `[__first,__last)` into the deque before the location specified by `__position`. This is known as *range insert*.

**insert()** [3/5]

```
template<typename _Tp, typename _Alloc>
deque< _Tp, _Alloc >::iterator deque::insert (
    const_iterator __position,
    const value_type & __x)
```

Inserts given value into deque before specified iterator.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__position</code> | A const_iterator into the deque. |
| <code>__x</code>        | Data to be inserted.             |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

References [deque\(\)](#), [insert\(\)](#), [push\\_back\(\)](#), and [push\\_front\(\)](#).

Referenced by [insert\(\)](#).



**insert()** [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
    const_iterator __position,
    size_type __n,
    const value_type & __x) [inline]
```

Inserts a number of copies of given data into the deque.

**Parameters**

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | A const_iterator into the deque.   |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

**Returns**

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by `__position`.

**insert()** [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
    const_iterator __position,
    value_type && __x) [inline]
```

Inserts given rvalue into deque before specified iterator.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__position</code> | A const_iterator into the deque. |
| <code>__x</code>        | Data to be inserted.             |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

**max\_size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::max_size () const [inline], [nodiscard], [noexcept]
```

Returns the size() of the largest possible deque.

Referenced by [\\_M\\_new\\_elements\\_at\\_back\(\)](#), [\\_M\\_new\\_elements\\_at\\_front\(\)](#), [\\_M\\_push\\_back\\_aux\(\)](#), [\\_M\\_push\\_front\\_aux\(\)](#), and [\\_M\\_reallocate\\_map\(\)](#).

**operator=()** [1/3]

```
template<typename _Tp, typename _Alloc>
deque< _Tp, _Alloc > & deque::operator= (
    const deque< _Tp, _Alloc > & __x)
```

Deque assignment operator.

## Parameters

|                                              |                                                   |
|----------------------------------------------|---------------------------------------------------|
| <a href="#"><math>\leftrightarrow</math></a> | A deque of identical element and allocator types. |
| <a href="#">__x</a>                          |                                                   |

All the elements of *x* are copied.

The newly-created deque uses a copy of the allocator object used by [\\_\\_x](#) (unless the allocator traits dictate a different object).

References [deque\(\)](#), [std::\\_\\_addressof\(\)](#), [begin\(\)](#), [end\(\)](#), [get\\_allocator\(\)](#), [size\(\)](#), and [std::size\(\)](#).

**operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque & std::deque<_Tp, _Alloc >::operator= (
    deque<_Tp, _Alloc > && __x) [inline], [noexcept]
```

Deque move assignment operator.

## Parameters

|                                              |                                                   |
|----------------------------------------------|---------------------------------------------------|
| <a href="#"><math>\leftrightarrow</math></a> | A deque of identical element and allocator types. |
| <a href="#">__x</a>                          |                                                   |

The contents of [\\_\\_x](#) are moved into this deque (without copying, if the allocators permit it). [\\_\\_x](#) is a valid, but unspecified deque.

**operator=()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque & std::deque<_Tp, _Alloc >::operator= (
    initializer_list<value_type> & __l) [inline]
```

Assigns an initializer list to a deque.

## Parameters

|                   |                      |
|-------------------|----------------------|
| <a href="#">↩</a> | An initializer_list. |
| <a href="#">↩</a> |                      |
| <a href="#">↩</a> |                      |
| <a href="#">↩</a> |                      |
| <a href="#">/</a> |                      |

This function fills a deque with copies of the elements in the initializer\_list [\\_\\_l](#).

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

**operator[]()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque<_Tp, _Alloc >::operator[] (
    size_type __n) const [inline], [nodiscard], [noexcept]
```

Subscript access to the data contained in the deque.

## Parameters

|                                              |                                                             |
|----------------------------------------------|-------------------------------------------------------------|
| <a href="#"><math>\leftrightarrow</math></a> | The index of the element for which data should be accessed. |
| <a href="#">__n</a>                          |                                                             |

### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

### `operator[]()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::operator[] (
    size_type __n) [inline], [nodiscard], [noexcept]
```

Subscript access to the data contained in the deque.

### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

### Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

### `pop_back()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_back () [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

### `pop_front()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_front () [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

### `push_back()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_back (
    const value_type & __x) [inline]
```

Add data to the end of the deque.

## Parameters

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | Data to be added. |
| <code>_x</code> |                   |

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Referenced by [\\_M\\_range\\_initialize\(\)](#), and [insert\(\)](#).

**push\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::push_front (
    const value_type & __x) [inline]
```

Add data to the front of the deque.

## Parameters

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | Data to be added. |
| <code>_x</code> |                   |

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Referenced by [insert\(\)](#).

**rbegin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque<_Tp, _Alloc>::rbegin () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

**rbegin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque<_Tp, _Alloc>::rbegin () [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

**rend()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque<_Tp, _Alloc>::rend () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

**rend()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque<_Tp, _Alloc>::rend () [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

**resize()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
void std::deque<_Tp, _Alloc>::resize (
    size_type __new_size) [inline]
```

Resizes the deque to the specified number of elements.

**Parameters**

|                         |                                              |
|-------------------------|----------------------------------------------|
| <code>__new_size</code> | Number of elements the deque should contain. |
|-------------------------|----------------------------------------------|

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

**resize()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
void std::deque<_Tp, _Alloc>::resize (
    size_type __new_size,
    const value_type & __x) [inline]
```

Resizes the deque to the specified number of elements.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the deque should contain.      |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

**shrink\_to\_fit()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
void std::deque<_Tp, _Alloc>::shrink_to_fit () [inline], [noexcept]
```

A non-binding request to reduce memory use.

**size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
size_type std::deque<_Tp, _Alloc>::size () const [inline], [nodiscard], [noexcept]
```

Returns the number of elements in the deque.

Referenced by `std::deque<_Tp, polymorphic_allocator<_Tp>>::M_range_check()`, `std::operator<=>()`, `operator=()`, `std::deque<_Tp, polymorphic_allocator<_Tp>>::resize()`, and `std::deque<_Tp, polymorphic_allocator<_Tp>>::resize()`.

**swap()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
void std::deque<_Tp, _Alloc>::swap (
    deque<_Tp, _Alloc> & __x) [inline], [noexcept]
```

Swaps data with another deque.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A deque of the same element and allocator types. |
|------------------|--------------------------------------------------|

---

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

The documentation for this class was generated from the following files:

- [stl\\_deque.h](#)
- [deque.tcc](#)

## 5.382 `std::destroying_delete_t` Struct Reference

```
#include <new>
```

### 5.382.1 Detailed Description

Tag type used to declare a class-specific operator delete that can invoke the destructor before deallocating the memory.

The documentation for this struct was generated from the following file:

- [new](#)

## 5.383 `std::tr2::direct_bases<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- typedef [\\_\\_reflection\\_typelist](#)< `__direct_bases(_Tp)...` > `type`

### 5.383.1 Detailed Description

```
template<typename _Tp>  
struct std::tr2::direct_bases<_Tp>
```

Enumerate all the direct base classes of a class. Form of a typelist.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 5.384 `__gnu_pbds::direct_mask_range_hashing<Size_Type>` Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::direct_mask_range_hashing< Size_Type >`:



### Public Types

- typedef `Size_Type` **size\_type**

### Public Member Functions

- void **swap** ([direct\\_mask\\_range\\_hashing](#)< `Size_Type` > &other)

### Protected Member Functions

- void **notify\_resized** (`size_type` size)
- `size_type` [operator\(\)](#) (`size_type` hash) const
- `size_type` **range\_hash** (`size_type` hash) const
- void **swap** ([mask\\_based\\_range\\_hashing](#) &other)

#### 5.384.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mask_range_hashing< Size_Type >
```

A mask range-hashing class (uses a bitmask).

#### 5.384.2 Member Function Documentation

##### **operator()()**

```
template<typename Size_Type>
direct\_mask\_range\_hashing< Size_Type >::size_type \_\_gnu\_pbds::direct\_mask\_range\_hashing< Size_Type >::operator() (
    size_type hash) const [inline], [protected]
```

Transforms the `__hash` value hash into a ranged-hash value (using a bit-mask).

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

**5.385 `__gnu_pbds::direct_mod_range_hashing< Size_Type >` Class Template Reference**

```
#include <hash_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::direct_mod_range_hashing< Size_Type >`:

**Public Types**

- typedef `Size_Type` **size\_type**

**Public Member Functions**

- void **swap** (`direct_mod_range_hashing< Size_Type > &other`)

**Protected Member Functions**

- void **notify\_resized** (`size_type s`)
- void **notify\_resized** (`size_type size`)
- `size_type` **operator()** (`size_type hash`) const
- `size_type` **range\_hash** (`size_type s`) const
- void **swap** (`mod_based_range_hashing &other`)

**5.385.1 Detailed Description**

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mod_range_hashing< Size_Type >
```

A mod range-hashing class (uses the modulo function).

**5.385.2 Member Function Documentation****`operator()`**

```
template<typename Size_Type>
direct_mod_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mod_range_hashing< Size_Type
>::operator() (
    size_type hash) const [inline], [protected]
```



Transforms the `__hash` value hash into a ranged-hash value (using a modulo operation).  
 The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 5.386 `std::filesystem::directory_entry` Class Reference

```
#include <filesystem>
```

### Public Member Functions

- `directory_entry` (const `directory_entry` &)=default
- `directory_entry` (const `filesystem::path` &\_\_p)
- `directory_entry` (const `filesystem::path` &\_\_p, `error_code` &\_\_ec)
- `directory_entry` (`directory_entry` &&) noexcept=default
- void `assign` (const `filesystem::path` &\_\_p)
- void `assign` (const `filesystem::path` &\_\_p, `error_code` &\_\_ec)
- bool `exists` () const
- bool `exists` (`error_code` &\_\_ec) const noexcept
- `uintmax_t` `file_size` () const
- `uintmax_t` `file_size` (`error_code` &\_\_ec) const noexcept
- `uintmax_t` `hard_link_count` () const
- `uintmax_t` `hard_link_count` (`error_code` &\_\_ec) const noexcept
- bool `is_block_file` () const
- bool `is_block_file` (`error_code` &\_\_ec) const noexcept
- bool `is_character_file` () const
- bool `is_character_file` (`error_code` &\_\_ec) const noexcept
- bool `is_directory` () const
- bool `is_directory` (`error_code` &\_\_ec) const noexcept
- bool `is_fifo` () const
- bool `is_fifo` (`error_code` &\_\_ec) const noexcept
- bool `is_other` () const
- bool `is_other` (`error_code` &\_\_ec) const noexcept
- bool `is_regular_file` () const
- bool `is_regular_file` (`error_code` &\_\_ec) const noexcept
- bool `is_socket` () const
- bool `is_socket` (`error_code` &\_\_ec) const noexcept
- bool `is_symlink` () const
- bool `is_symlink` (`error_code` &\_\_ec) const noexcept
- `file_time_type` `last_write_time` () const
- `file_time_type` `last_write_time` (`error_code` &\_\_ec) const noexcept
- `operator` const `filesystem::path` & () const noexcept
- strong\_ordering `operator<=>` (const `directory_entry` &\_\_rhs) const noexcept
- `directory_entry` & `operator=` (const `directory_entry` &)=default
- `directory_entry` & `operator=` (`directory_entry` &&) noexcept=default
- bool `operator==` (const `directory_entry` &\_\_rhs) const noexcept
- const `filesystem::path` & `path` () const noexcept
- void `refresh` ()
- void `refresh` (`error_code` &\_\_ec) noexcept
- void `replace_filename` (const `filesystem::path` &\_\_p)
- void `replace_filename` (const `filesystem::path` &\_\_p, `error_code` &\_\_ec)
- `file_status` `status` () const
- `file_status` `status` (`error_code` &\_\_ec) const noexcept
- `file_status` `symlink_status` () const
- `file_status` `symlink_status` (`error_code` &\_\_ec) const noexcept

## Friends

- struct **\_Dir**
- class **directory\_iterator**
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [directory\\_entry](#) &\_\_d)
- class **recursive\_directory\_iterator**

### 5.386.1 Detailed Description

The value type used by directory iterators.

Since

C++17

The documentation for this class was generated from the following file:

- [bits/fs\\_dir.h](#)

## 5.387 std::filesystem::directory\_iterator Class Reference

```
#include <filesystem>
```

## Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [input\\_iterator\\_tag](#) **iterator\_category**
- typedef const [directory\\_entry](#) \* **pointer**
- typedef const [directory\\_entry](#) & **reference**
- typedef [directory\\_entry](#) **value\_type**

## Public Member Functions

- **directory\_iterator** (const [directory\\_iterator](#) &\_\_rhs)=default
- **directory\_iterator** (const [path](#) &\_\_p)
- **directory\_iterator** (const [path](#) &\_\_p, [directory\\_options](#) \_\_options)
- **directory\_iterator** (const [path](#) &\_\_p, [directory\\_options](#) \_\_options, [error\\_code](#) &\_\_ec)
- **directory\_iterator** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- **directory\_iterator** ([directory\\_iterator](#) &&\_\_rhs) noexcept=default
- [directory\\_iterator](#) & **increment** ([error\\_code](#) &\_\_ec)
- const [directory\\_entry](#) & **operator**\* () const noexcept
- [directory\\_iterator](#) & **operator++** ()
- \_\_directory\_iterator\_proxy **operator++** (int)
- const [directory\\_entry](#) \* **operator->** () const noexcept
- [directory\\_iterator](#) & **operator=** (const [directory\\_iterator](#) &\_\_rhs)=default
- [directory\\_iterator](#) & **operator=** ([directory\\_iterator](#) &&\_\_rhs) noexcept=default
- bool **operator==** ([default\\_sentinel\\_t](#)) const noexcept

## Friends

- bool **operator==** (const [directory\\_iterator](#) &\_\_lhs, const [directory\\_iterator](#) &\_\_rhs) noexcept
- class **recursive\_directory\_iterator**

## Related Symbols

(Note that these are not member symbols.)

- [directory\\_iterator begin](#) ([directory\\_iterator](#) \_\_iter) noexcept
- [directory\\_iterator end](#) ([directory\\_iterator](#)) noexcept
  
- [recursive\\_directory\\_iterator begin](#) ([recursive\\_directory\\_iterator](#) \_\_iter) noexcept
- [recursive\\_directory\\_iterator end](#) ([recursive\\_directory\\_iterator](#)) noexcept

### 5.387.1 Detailed Description

Iterator type for traversing the entries in a single directory.

Since

C++17

The documentation for this class was generated from the following file:

- [bits/fs\\_dir.h](#)

## 5.388 `std::discard_block_engine<_RandomNumberEngine, __p, __r>` Class Template Reference

```
#include <random>
```

### Public Types

- `template<typename _Sseq>`  
`using \_If\_seed\_seq`
- `typedef _RandomNumberEngine::result_type result\_type`

### Public Member Functions

- [discard\\_block\\_engine](#) ()
- [discard\\_block\\_engine](#) ([\\_RandomNumberEngine](#) &&\_\_rng)
- `template<typename _Sseq, typename = \_If\_seed\_seq<_Sseq>>`  
[discard\\_block\\_engine](#) ([\\_Sseq](#) &\_\_q)
- [discard\\_block\\_engine](#) (const [\\_RandomNumberEngine](#) &\_\_rng)
- [discard\\_block\\_engine](#) ([result\\_type](#) \_\_s)
- const [\\_RandomNumberEngine](#) & [base](#) () const noexcept
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator\(\)](#) ()
- void [seed](#) ()
- `template<typename _Sseq>`  
`\_If\_seed\_seq<_Sseq> seed (\_Sseq &__q)`
- void [seed](#) ([result\\_type](#) \_\_s)

### Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

### Static Public Attributes

- static constexpr size\_t **block\_size**
- static constexpr size\_t **used\_block**

### Friends

- template<typename \_RandomNumberEngine1, size\_t \_\_p1, size\_t \_\_r1, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const std::discard\_block\_engine< \_RandomNumberEngine1, \_\_p1, \_\_r1 > &\_\_x)
- bool operator== (const discard\_block\_engine &\_\_lhs, const discard\_block\_engine &\_\_rhs)
- template<typename \_RandomNumberEngine1, size\_t \_\_p1, size\_t \_\_r1, typename \_CharT, typename \_Traits>  
std::basic\_istream< \_CharT, \_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is, std::discard\_block\_engine< \_RandomNumberEngine1, \_\_p1, \_\_r1 > &\_\_x)

#### 5.388.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
class std::discard_block_engine< _RandomNumberEngine, __p, __r >
```

Produces random numbers from some base engine by discarding blocks of data.

Precondition

$$0 \leq r \leq p$$

Since

C++11

#### 5.388.2 Member Typedef Documentation

##### result\_type

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
typedef _RandomNumberEngine::result_type std::discard_block_engine< _RandomNumberEngine, __p, __r
>::result_type
```

The type of the generated random value.

#### 5.388.3 Constructor & Destructor Documentation

##### discard\_block\_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine () [inline]
```

Constructs a default discard\_block\_engine engine.

The underlying engine is default constructed as well.

##### discard\_block\_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
    const _RandomNumberEngine & __rng) [inline], [explicit]
```

Copy constructs a discard\_block\_engine engine.

Copies an existing base class random number generator.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

**discard\_block\_engine() [3/5]**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
    _RandomNumberEngine && __rng) [inline], [explicit]
```

Move constructs a `discard_block_engine` engine.  
Copies an existing base class random number generator.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

**discard\_block\_engine() [4/5]**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
    result_type __s) [inline], [explicit]
```

Seed constructs a `discard_block_engine` engine.  
Constructs the underlying generator engine seeded with `__s`.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A seed value for the base class engine. |
|------------------|-----------------------------------------|

**discard\_block\_engine() [5/5]**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
    _Sseq & __q) [inline], [explicit]
```

Generator construct a `discard_block_engine` engine.

**Parameters**

|                  |                  |
|------------------|------------------|
| <code>__q</code> | A seed sequence. |
|------------------|------------------|

**5.388.4 Member Function Documentation****base()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
const _RandomNumberEngine & std::discard_block_engine< _RandomNumberEngine, __p, __r >::base ()
const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

**discard()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard (
    unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

**max()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
static constexpr result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::max ()
[inline], [static], [constexpr]
```

Gets the maximum value in the generated random number range.

**min()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
static constexpr result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::min ()
[inline], [static], [constexpr]
```

Gets the minimum value in the generated random number range.

**operator()()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
discard_block_engine<_RandomNumberEngine, __p, __r>::result_type std::discard_block_engine<_↵
RandomNumberEngine, __p, __r>::operator() ()
```

Gets the next value in the generated random number sequence.

**seed()** [1/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed () [inline]
```

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

**seed()** [2/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq>
_If_seed_seq<_Sseq> std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed (
    _Sseq & __q) [inline]
```

Reseeds the discard\_block\_engine object with the given seed sequence.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <b><code>_↵</code></b> | A seed generator function. |
| <b><code>_q</code></b> |                            |

**seed()** [3/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed (
    result_type __s) [inline]
```

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

### 5.388.5 Friends And Related Symbol Documentation

#### operator<<

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
```

```
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x) [friend]
```

Inserts the current state of a discard\_block\_engine random number generator engine \_\_x into the output stream \_\_os.

#### Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A discard_block_engine random number generator engine. |

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

#### operator==

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool operator== (
    const discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs,
    const discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs) [friend]
```

Compares two discard\_block\_engine random number generator objects of the same type for equality.

#### Parameters

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>__lhs</code> | A discard_block_engine random number generator object.       |
| <code>__rhs</code> | Another discard_block_engine random number generator object. |

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

#### operator>>

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
```

```
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

#### Parameters

|                   |                  |
|-------------------|------------------|
| <code>__is</code> | An input stream. |
|-------------------|------------------|

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <a href="#">↩</a>  | A discard_block_engine random number generator engine. |
| <a href="#">_X</a> |                                                        |

### Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.389 std::discrete\_distribution< \_IntType > Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_IntType` [result\\_type](#)

### Public Member Functions

- template<typename `_InputIterator`>  
**discrete\_distribution** (`_InputIterator` \_\_wbegin, `_InputIterator` \_\_wend)
- **discrete\_distribution** (const [param\\_type](#) &\_\_p)
- **discrete\_distribution** ([initializer\\_list](#)< double > \_\_wl)
- template<typename `_Func`>  
**discrete\_distribution** (size\_t \_\_nw, double \_\_xmin, double \_\_xmax, `_Func` \_\_fw)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **\_\_generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **\_\_generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator`>  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- **std::vector**< double > **probabilities** () const
- void **reset** ()



## Friends

- `template<typename _IntType1, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::discrete_distribution< _IntType1 > &__x)`
- `bool operator== (const discrete_distribution &__d1, const discrete_distribution &__d2)`
- `template<typename _IntType1, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::discrete_distribution< _IntType1 > &__x)`

### 5.389.1 Detailed Description

```
template<typename _IntType = int>
class std::discrete_distribution< _IntType >
```

A `discrete_distribution` random number distribution.

This distribution produces random numbers  $i, 0 \leq i < n$ , distributed according to the probability mass function  $p(i|p_0, \dots, p_{n-1}) = p_i$ .

Since

C++11

### 5.389.2 Member Typedef Documentation

#### result\_type

```
template<typename _IntType = int>
typedef _IntType std::discrete_distribution< _IntType >::result_type
```

The type of the range of the distribution.

### 5.389.3 Member Function Documentation

#### max()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::discrete_distribution< _IntType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by `operator>()()`.

**param()** [1/2]

```
template<typename _IntType = int>
param_type std::discrete_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

**param()** [2/2]

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**probabilities()**

```
template<typename _IntType = int>
std::vector< double > std::discrete_distribution< _IntType >::probabilities () const [inline]
```

Returns the probabilities of the distribution.

**reset()**

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

**5.389.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::discrete_distribution< _IntType1 > & __x) [friend]
```

Inserts a discrete\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <code>__os</code> | An output stream.                                   |
| <code>__x</code>  | A discrete_distribution random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _IntType = int>
bool operator== (
    const discrete\_distribution< _IntType > & __d1,
    const discrete\_distribution< _IntType > & __d2) [friend]
```

Return true if two discrete distributions have the same parameters.

**operator>>**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic\_istream< _CharT, _Traits > & operator>> (
    std::basic\_istream< _CharT, _Traits > & __is,
    std::discrete\_distribution< _IntType1 > & __x) [friend]
```

Extracts a [discrete\\_distribution](#) random number distribution \_\_x from the input stream \_\_is.

**Parameters**

|                      |                                                                         |
|----------------------|-------------------------------------------------------------------------|
| <a href="#">__is</a> | An input stream.                                                        |
| <a href="#">__x</a>  | A <a href="#">discrete_distribution</a> random number generator engine. |

**Returns**

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.390 std::divides< \_Tp > Struct Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for [std::divides< \\_Tp >](#):



**Public Types**

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

**Public Member Functions**

- constexpr `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

**5.390.1 Detailed Description**

**template<typename `_Tp`>**  
**struct `std::divides<_Tp>`**

One of the [math functors](#).

**5.390.2 Member Typedef Documentation****first\_argument\_type**

typedef `_Tp` [std::binary\\_function<\\_Tp, \\_Tp, \\_Tp>::first\\_argument\\_type](#) [inherited]  
`first_argument_type` is the type of the first argument

**result\_type**

typedef `_Tp` [std::binary\\_function<\\_Tp, \\_Tp, \\_Tp>::result\\_type](#) [inherited]  
`result_type` is the return type

**second\_argument\_type**

typedef `_Tp` [std::binary\\_function<\\_Tp, \\_Tp, \\_Tp>::second\\_argument\\_type](#) [inherited]  
`second_argument_type` is the type of the second argument  
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

**5.391 std::divides< void > Struct Reference**

`#include <stl_function.h>`

Inheritance diagram for `std::divides< void >`:



## Public Types

- typedef void [first\\_argument\\_type](#)
- typedef \_\_is\_transparent [is\\_transparent](#)
- typedef void [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

## Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t)/[std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t)/[std::forward](#)< \_Up >(\_\_u))
- constexpr void **operator()** (const void &\_\_x, const void &\_\_y) const

### 5.391.1 Detailed Description

One of the [math functors](#).

### 5.391.2 Member Typedef Documentation

#### **first\_argument\_type**

typedef void [std::binary\\_function](#)< void, void, void >::first\_argument\_type  
first\_argument\_type is the type of the first argument

#### **result\_type**

typedef void [std::binary\\_function](#)< void, void, void >::result\_type  
result\_type is the return type

#### **second\_argument\_type**

typedef void [std::binary\\_function](#)< void, void, void >::second\_argument\_type  
second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.392 std::domain\_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for `std::domain_error`:



### Public Member Functions

- **domain\_error** (const char \*)
- **domain\_error** (const [domain\\_error](#) &)=default
- **domain\_error** (const [string](#) &\_\_arg)
- **domain\_error** ([domain\\_error](#) &&)=default
- **domain\_error** & **operator=** (const [domain\\_error](#) &)=default
- **domain\_error** & **operator=** ([domain\\_error](#) &&)=default
- virtual const char \* **what** () const noexcept

#### 5.392.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

#### 5.392.2 Member Function Documentation

##### **what()**

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 5.393 `__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >` Struct Template Reference

```
#include <null_node_metadata.hpp>
```

## Public Types

- typedef const\_iterator **const\_reference**
- typedef const\_reference **reference**
- typedef const\_iterator **value\_type**

### 5.393.1 Detailed Description

```
template<typename Key, typename Data, typename _Alloc>
struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >
```

Constant node iterator.

The documentation for this struct was generated from the following file:

- `null_node_metadata.hpp`

## 5.394 std::chrono::duration<\_Rep, \_Period> Class Template Reference

## Public Types

- using **period**
- using **rep**

## Public Member Functions

- `template<typename _Rep2, typename = _Require< is_convertible<const _Rep2&, rep>, __or_<__is_float<rep>, __not_<__is_float<__Rep2>>>>>>`  
`constexpr duration (const _Rep2 &__rep)`
- `duration (const duration &)=default`
- `template<typename _Rep2, typename _Period2, typename = _Require< is_convertible<const _Rep2&, rep>, __or_<__is_float<rep>, __and_<__is_harmonic<_Period2>, __not_<__is_float<_Rep2>>>>>>>`  
`constexpr duration (const duration< _Rep2, _Period2 > &__d)`
- `constexpr rep count () const`
- `template<typename _Rep2 = rep>`  
`constexpr __enable_if_t<!treat_as_floating_point< _Rep2 >::value, duration & > operator%= (const duration &__d)`
- `template<typename _Rep2 = rep>`  
`constexpr __enable_if_t<!treat_as_floating_point< _Rep2 >::value, duration & > operator%= (const rep &__rhs)`
- `constexpr duration & operator*= (const rep &__rhs)`
- `constexpr duration< typename common_type< rep >::type, period > operator+ () const`
- `constexpr duration & operator++ ()`
- `constexpr duration operator++ (int)`
- `constexpr duration & operator+= (const duration &__d)`
- `constexpr duration< typename common_type< rep >::type, period > operator- () const`
- `constexpr duration & operator-- ()`
- `constexpr duration operator-- (int)`
- `constexpr duration & operator-= (const duration &__d)`
- `constexpr duration & operator/= (const rep &__rhs)`
- `duration & operator= (const duration &)=default`

## Static Public Member Functions

- static constexpr **duration** **max** () noexcept
- static constexpr **duration** **min** () noexcept
- static constexpr **duration** **zero** () noexcept

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+`  
`(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2>`  
`constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator*` `(const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool operator==` `(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`

### 5.394.1 Detailed Description

`template<typename _Rep, typename _Period>`  
**class** `std::chrono::duration<_Rep, _Period>`

`std::chrono::duration` represents a distance between two points in time

The documentation for this class was generated from the following file:

- [chrono.h](#)

## 5.395 std::chrono::duration\_values<\_Rep> Struct Template Reference

```
#include <chrono.h>
```

### Static Public Member Functions

- `static constexpr _Rep max () noexcept`
- `static constexpr _Rep min () noexcept`
- `static constexpr _Rep zero () noexcept`

### 5.395.1 Detailed Description

`template<typename _Rep>`  
**struct** `std::chrono::duration_values<_Rep>`

`duration_values`

The documentation for this struct was generated from the following file:

- [chrono.h](#)

## 5.396 std::tr2::dynamic\_bitset<\_WordT, \_Alloc> Class Template Reference

```
#include <dynamic_bitset>
```



Inheritance diagram for `std::tr2::dynamic_bitset< _WordT, _Alloc >`:



## Classes

- class [reference](#)

## Public Types

- typedef `__dynamic_bitset_base< _WordT, _Alloc > _Base`
- typedef `_Alloc allocator_type`
- typedef `_WordT block_type`
- typedef `bool const_reference`
- typedef `size_t size_type`

## Public Member Functions

- `dynamic_bitset` ()=default
- `dynamic_bitset` (const allocator\_type &\_\_alloc)
- `dynamic_bitset` (const char \* \_\_str, const allocator\_type &\_\_alloc=allocator\_type())
- `dynamic_bitset` (const `dynamic_bitset` &)=default
- template<typename \_CharT, typename \_Traits, typename \_Alloc1>  
`dynamic_bitset` (const `std::basic_string`< \_CharT, \_Traits, \_Alloc1 > &\_\_str, typename `basic_string`< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_pos=0, typename `basic_string`< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_n=`std::basic_string`< \_CharT, \_Traits, \_Alloc1 >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'), const allocator\_type &\_\_alloc=allocator\_type())
- `dynamic_bitset` (`dynamic_bitset` &&\_\_b) noexcept
- `dynamic_bitset` (`initializer_list`< block\_type > \_\_il, const allocator\_type &\_\_alloc=allocator\_type())
- `dynamic_bitset` (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type &\_\_alloc=allocator\_type())
- template<typename \_Traits = std::char\_traits<char>, typename \_CharT = typename \_Traits::char\_type>  
void `_M_copy_from_ptr` (const \_CharT \*, size\_t, size\_t, size\_t, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc1>  
void `_M_copy_from_string` (const `basic_string`< \_CharT, \_Traits, \_Alloc1 > &\_\_str, size\_t \_\_pos, size\_t \_\_n, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))

- template<typename \_CharT, typename \_Traits, typename \_Alloc1>  
void **\_M\_copy\_to\_string** (std::basic\_string<\_CharT, \_Traits, \_Alloc1 > &\_\_str, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1')) const
- bool **all** () const
- bool **any** () const
- template<typename \_BlockInputIterator>  
void **append** (\_BlockInputIterator \_\_first, \_BlockInputIterator \_\_last)
- void **append** (block\_type \_\_block)
- void **append** (initializer\_list< block\_type > \_\_il)
- void **clear** ()
- size\_type **count** () const noexcept
- bool **empty** () const noexcept
- size\_type **find\_first** () const
- size\_type **find\_next** (size\_t \_\_prev) const
- **dynamic\_bitset** & **flip** ()
- **dynamic\_bitset** & **flip** (size\_type \_\_pos)
- allocator\_type **get\_allocator** () const noexcept
- bool **is\_proper\_subset\_of** (const **dynamic\_bitset** &\_\_b) const
- bool **is\_subset\_of** (const **dynamic\_bitset** &\_\_b) const
- constexpr size\_type **max\_size** () noexcept
- bool **none** () const
- size\_type **num\_blocks** () const noexcept
- **dynamic\_bitset** & **operator=** (const **dynamic\_bitset** &)=default
- **dynamic\_bitset** & **operator=** (**dynamic\_bitset** &&\_\_b) noexcept(std::is\_nothrow\_move\_assignable<\_Base >::value)
- **dynamic\_bitset** **operator~** () const
- void **push\_back** (bool \_\_bit)
- **dynamic\_bitset** & **reset** ()
- **dynamic\_bitset** & **reset** (size\_type \_\_pos)
- void **resize** (size\_type \_\_nbits, bool \_\_value=false)
- **dynamic\_bitset** & **set** ()
- **dynamic\_bitset** & **set** (size\_type \_\_pos, bool \_\_val=true)
- size\_type **size** () const noexcept
- void **swap** (**dynamic\_bitset** &\_\_b) noexcept
- bool **test** (size\_type \_\_pos) const
- template<typename \_CharT = char, typename \_Traits = std::char\_traits<\_CharT>, typename \_Alloc1 = std::allocator<\_CharT>>  
std::basic\_string<\_CharT, \_Traits, \_Alloc1 > **to\_string** (\_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1')) const
- unsigned long long **to\_ullong** () const
- unsigned long **to\_ulong** () const
- **dynamic\_bitset** & **operator&=** (const **dynamic\_bitset** &\_\_rhs)
- **dynamic\_bitset** & **operator&=** (**dynamic\_bitset** &&\_\_rhs)
- **dynamic\_bitset** & **operator|=** (const **dynamic\_bitset** &\_\_rhs)
- **dynamic\_bitset** & **operator^=** (const **dynamic\_bitset** &\_\_rhs)
- **dynamic\_bitset** & **operator-=** (const **dynamic\_bitset** &\_\_rhs)
- **dynamic\_bitset** & **operator<<=** (size\_type \_\_pos)
- **dynamic\_bitset** & **operator>>=** (size\_type \_\_pos)

- [reference operator\[\]](#) (size\_type \_\_pos)
- [const\\_reference operator\[\]](#) (size\_type \_\_pos) const

- [dynamic\\_bitset operator<<](#) (size\_type \_\_pos) const
- [dynamic\\_bitset operator>>](#) (size\_type \_\_pos) const

### Static Public Attributes

- static const size\_type **bits\_per\_block**
- static const size\_type **npos**

### Friends

- bool **operator<** (const [dynamic\\_bitset](#) &\_\_lhs, const [dynamic\\_bitset](#) &\_\_rhs) noexcept
- bool **operator==** (const [dynamic\\_bitset](#) &\_\_lhs, const [dynamic\\_bitset](#) &\_\_rhs) noexcept
- class **reference**

### 5.396.1 Detailed Description

**template<typename \_WordT = unsigned long long, typename \_Alloc = std::allocator<\_WordT>>**  
**class std::tr2::dynamic\_bitset<\_WordT, \_Alloc >**

The `dynamic_bitset` class represents a sequence of bits.

See N2050, Proposal to Add a Dynamically Sizeable Bitset to the Standard Library. <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2050.pdf>

In the general unoptimized case, storage is allocated in word-sized blocks. Let B be the number of bits in a word, then  $(Nb+(B-1))/B$  words will be used for storage. B - NbB bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `dynamic_bitset` as "a simple array of bits," be aware that your mental picture is reversed: a `dynamic_bitset` behaves the same way as bits in integers do, with the bit at index 0 in the "least significant / right-hand" position, and the bit at index Nb-1 in the "most significant / left-hand" position. Thus, unlike other containers, a `dynamic_bitset`'s index "counts from right to left," to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints "b('a') is 0001100001" on a modern ASCII system.

```
#include <dynamic_bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long    a = 'a';
    dynamic_bitset<> b(a);

    cout << "b('a') is " << b << endl;

    ostringstream s;
    s << b;
    string str = s.str();
    cout << "index 3 in the string is " << str[3] << " but\n"
         << "index 3 in the bitset is " << b[3] << endl;
}
```

Most of the actual code isn't contained in `dynamic_bitset<>` itself, but in the base class `__dynamic_bitset_base`. The base class works with whole words, not with individual bits. This allows us to specialize `__dynamic_bitset_base` for the important special case where the `dynamic_bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `__dynamic_bitset_base` *is* a vector, and is indexed as such. This is carefully encapsulated.

### 5.396.2 Constructor & Destructor Documentation

#### dynamic\_bitset() [1/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset () [default]
All bits set to zero.
```

#### dynamic\_bitset() [2/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    const allocator_type & __alloc) [inline], [explicit]
All bits set to zero.
```

#### dynamic\_bitset() [3/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    size_type __nbits,
    unsigned long long __val = 0ULL,
    const allocator_type & __alloc = allocator_type()) [inline], [explicit]
Initial bits bitwise-copied from a single word (others set to zero).
```

#### dynamic\_bitset() [4/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT, typename _Traits, typename _Alloc1>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    const std::basic_string< _CharT, _Traits, _Alloc1 > & __str,
    typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos = 0,
    typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n = std::basic_string<←
    _CharT, _Traits, _Alloc1>::npos,
    _CharT __zero = _CharT('0'),
    _CharT __one = _CharT('1'),
    const allocator_type & __alloc = allocator_type()) [inline], [explicit]
Use a subset of a string.
```

#### Parameters

|                      |                                                            |
|----------------------|------------------------------------------------------------|
| <code>__str</code>   | A string of '0' and '1' characters.                        |
| <code>__pos</code>   | Index of the first character in <code>__str</code> to use. |
| <code>__n</code>     | The number of characters to copy.                          |
| <code>__zero</code>  | The character to use for unset bits.                       |
| <code>__one</code>   | The character to use for set bits.                         |
| <code>__alloc</code> | An allocator.                                              |

#### Exceptions

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| <code>std::out_of_range</code>     | If <code>__pos</code> is bigger the size of <code>__str</code> .   |
| <code>std::invalid_argument</code> | If a character appears in the string which is neither '0' nor '1'. |

**dynamic\_bitset()** [5/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    const char * __str,
    const allocator_type & __alloc = allocator_type()) [inline], [explicit]
```

Construct from a string.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__str</code>   | A string of '0' and '1' characters. |
| <code>__alloc</code> | An allocator.                       |

**Exceptions**

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| <code>std::invalid_argument</code> | If a character appears in the string which is neither '0' nor '1'. |
|------------------------------------|--------------------------------------------------------------------|

**dynamic\_bitset()** [6/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    const dynamic_bitset< _WordT, _Alloc > & ) [default]
```

Copy constructor.

**dynamic\_bitset()** [7/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    dynamic_bitset< _WordT, _Alloc > && __b) [inline], [noexcept]
```

Move constructor.

**5.396.3 Member Function Documentation****all()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::all () const [inline]
```

Tests whether all the bits are on.

**Returns**

True if all the bits are set.

**any()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::any () const [inline]
```

Tests whether any of the bits are on.

**Returns**

True if at least one bit is set.

**append()** [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _BlockInputIterator>
void std::tr2::dynamic_bitset<_WordT, _Alloc>::append (
    _BlockInputIterator __first,
    _BlockInputIterator __last) [inline]
```

Append an iterator range of blocks.

**append()** [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset<_WordT, _Alloc>::append (
    block_type __block) [inline]
```

Append a block.

**clear()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset<_WordT, _Alloc>::clear () [inline]
```

Clear the bitset.

**count()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset<_WordT, _Alloc>::count () const [inline], [noexcept]
```

Returns the number of bits which are set.

**empty()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset<_WordT, _Alloc>::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the `dynamic_bitset` is empty.

**find\_first()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset<_WordT, _Alloc>::find_first () const [inline]
```

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or `size()` if not found.

See also

`find_next`

**find\_next()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset<_WordT, _Alloc>::find_next (
    size_t __prev) const [inline]
```

Finds the index of the next "on" bit after `prev`.

Returns

The index of the next bit set, or `size()` if not found.

#### Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>__prev</code> | Where to start searching. |
|---------------------|---------------------------|

#### See also

`find_first`

#### `flip()` [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::flip () [inline]
```

Toggles every bit to its opposite value.

#### `flip()` [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::flip (
    size_type __pos) [inline]
```

Toggles a given bit to its opposite value.

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__pos</code> | The index of the bit. |
|--------------------|-----------------------|

#### Exceptions

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

#### `get_allocator()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
allocator_type std::tr2::dynamic_bitset< _WordT, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Return the allocator for the bitset.

#### `max_size()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::max_size () [inline], [constexpr], [noexcept]
```

Returns the maximum size of a `dynamic_bitset` object having the same type as `*this`. The real answer is `max() * bits_per_block` but is likely to overflow.

#### `none()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::none () const [inline]
```

Tests whether any of the bits are on.

#### Returns

True if none of the bits are set.

**num\_blocks()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::num_blocks () const [inline], [noexcept]
```

Returns the total number of blocks.

**operator&=()** [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

**operator&=()** [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (
    dynamic_bitset< _WordT, _Alloc > && __rhs) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

**operator-=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator-= (
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

**operator<<()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<< (
    size_type __pos) const [inline]
```

Self-explanatory.



**operator<<=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<<= (
    size_type __pos) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | The number of places to shift. |
|--------------------|--------------------------------|

These should be self-explanatory.

**operator=() [1/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
    const dynamic_bitset< _WordT, _Alloc > & ) [default]
```

Copy assignment operator.

**operator=() [2/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
    dynamic_bitset< _WordT, _Alloc > && __b) [inline], [noexcept]
```

Move assignment operator.

**operator>>()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>> (
    size_type __pos) const [inline]
```

Self-explanatory.

**operator>>=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>>= (
    size_type __pos) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | The number of places to shift. |
|--------------------|--------------------------------|

These should be self-explanatory.

**operator[]() [1/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
    size_type __pos) [inline]
```

Array-indexing support.

**Parameters**

|                    |                                              |
|--------------------|----------------------------------------------|
| <code>__pos</code> | Index into the <code>dynamic_bitset</code> . |
|--------------------|----------------------------------------------|

**Returns**

A bool for a 'const `dynamic_bitset`'. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

**`operator[]()` [2/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
const_reference std::tr2::dynamic_bitset<_WordT, _Alloc >::operator[] (
    size_type __pos) const [inline]
```

Array-indexing support.

**Parameters**

|                    |                                              |
|--------------------|----------------------------------------------|
| <code>__pos</code> | Index into the <code>dynamic_bitset</code> . |
|--------------------|----------------------------------------------|

**Returns**

A bool for a 'const `dynamic_bitset`'. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

**`operator^=()`**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset<_WordT, _Alloc >::operator^= (
    const dynamic_bitset<_WordT, _Alloc > & __rhs) [inline]
```

Operations on `dynamic_bitsets`.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

**`operator" |=()`**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset<_WordT, _Alloc >::operator|= (
    const dynamic_bitset<_WordT, _Alloc > & __rhs) [inline]
```

Operations on `dynamic_bitsets`.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

**operator~()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator~ () const [inline]
```

See the no-argument `flip()`.

**push\_back()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back (
    bool __bit) [inline]
```

Push a bit onto the high end of the bitset.

**reset() [1/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::reset () [inline]
```

Sets every bit to false.

**reset() [2/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::reset (
    size_type __pos) [inline]
```

Sets a given bit to false.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__pos</code> | The index of the bit. |
|--------------------|-----------------------|

**Exceptions**

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

Same as writing `set (__pos, false)`.

**resize()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::resize (
    size_type __nbits,
    bool __value = false) [inline]
```

Resize the bitset.

Referenced by `std::tr2::operator>>()`.

**set()** [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::set () [inline]
```

Sets every bit to true.

**set()** [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::set (
    size_type __pos,
    bool __val = true) [inline]
```

Sets a given bit to a particular value.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__pos</code> | The index of the bit.                   |
| <code>__val</code> | Either true or false, defaults to true. |

**Exceptions**

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

**size()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::size () const [inline], [noexcept]
```

Returns the total number of bits.

Referenced by [std::tr2::operator>>\(\)](#).

**swap()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::swap (
    dynamic_bitset< _WordT, _Alloc > & __b) [inline], [noexcept]
```

Swap with another bitset.

**test()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::test (
    size_type __pos) const [inline]
```

Tests the value of a bit.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__pos</code> | The index of a bit. |
|--------------------|---------------------|

**Returns**

The value at `__pos`.

**Exceptions**

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

**to\_string()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 =
std::allocator<_CharT>>
std::basic_string<_CharT, _Traits, _Alloc1 > std::tr2::dynamic_bitset<_WordT, _Alloc >::to_↵
string (
    _CharT __zero = _CharT('0'),
    _CharT __one = _CharT('1')) const [inline]
```

Returns a character interpretation of the `dynamic_bitset`.

**Returns**

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

**to\_ulong()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long long std::tr2::dynamic_bitset<_WordT, _Alloc >::to_ulong () const [inline]
```

Returns a numerical interpretation of the `dynamic_bitset`.

**Returns**

The integral equivalent of the bits.

**Exceptions**

|                                  |                                                                   |
|----------------------------------|-------------------------------------------------------------------|
| <code>std::overflow_error</code> | If there are too many bits to be represented in an unsigned long. |
|----------------------------------|-------------------------------------------------------------------|

**to\_ulong()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long std::tr2::dynamic_bitset<_WordT, _Alloc >::to_ulong () const [inline]
```

Returns a numerical interpretation of the `dynamic_bitset`.

**Returns**

The integral equivalent of the bits.

**Exceptions**

|                                  |                                                                   |
|----------------------------------|-------------------------------------------------------------------|
| <code>std::overflow_error</code> | If there are too many bits to be represented in an unsigned long. |
|----------------------------------|-------------------------------------------------------------------|

The documentation for this class was generated from the following files:

- [dynamic\\_bitset](#)
- [dynamic\\_bitset.tcc](#)

## 5.397 `std::enable_if< bool, _Tp >` Struct Template Reference

```
#include <type_traits>
```

### 5.397.1 Detailed Description

```
template<bool, typename _Tp = void>
struct std::enable_if< bool, _Tp >
```

Define a member typedef `type` only if a boolean constant is true.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.398 `std::enable_shared_from_this< _Tp >` Class Template Reference

```
#include <memory>
```

### Public Member Functions

- [shared\\_ptr< \\_Tp > shared\\_from\\_this \(\)](#)
- [shared\\_ptr< const \\_Tp > shared\\_from\\_this \(\) const](#)

### Protected Member Functions

- [enable\\_shared\\_from\\_this](#) (const [enable\\_shared\\_from\\_this](#) &) noexcept
- [enable\\_shared\\_from\\_this](#) & [operator=](#) (const [enable\\_shared\\_from\\_this](#) &) noexcept

### Friends

- const [enable\\_shared\\_from\\_this](#) \* [\\_\\_enable\\_shared\\_from\\_this\\_base](#) (const [\\_\\_shared\\_count<>](#) &, const [enable\\_shared\\_from\\_this](#) \*\_\_p)
- template<typename, \_Lock\_policy>  
class [\\_\\_shared\\_ptr](#)

### 5.398.1 Detailed Description

```
template<typename _Tp>
class std::enable_shared_from_this< _Tp >
```

Base class allowing use of the member function `shared_from_this`.

Since

C++11

The documentation for this class was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 5.399 `__gnu_cxx::enc_filebuf< _CharT >` Class Template Reference

```
#include <enc_filebuf.h>
```

Inheritance diagram for `__gnu_cxx::enc_filebuf<_CharT>`:



## Public Types

- typedef `codecvt< char_type, char, __state_type >` **\_\_codecvt\_type**
- typedef `__basic_file< char >` **\_\_file\_type**
- typedef `basic_filebuf< char_type, traits_type >` **\_\_filebuf\_type**
- typedef `traits_type::state_type` **\_\_state\_type**
- typedef `basic_streambuf< char_type, traits_type >` **\_\_streambuf\_type**
- typedef `_CharT` **char\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::off_type` **off\_type**
- typedef `traits_type::pos_type` **pos\_type**
- typedef `traits_type::state_type` **state\_type**
- typedef `encoding_char_traits< _CharT >` **traits\_type**

## Public Member Functions

- **enc\_filebuf** (`state_type &__state`)
- `__filebuf_type * close` ()
- locale `getloc` () const
- streamsize `in_avail` ()
- bool `is_open` () const throw ()
- `_If_fs_path< _Path, __filebuf_type * >` `open` (const `_Path &__s`, `ios_base::openmode __mode`)
- `__filebuf_type * open` (const `char *__s`, `ios_base::openmode __mode`)
- `__filebuf_type * open` (const `std::string &__s`, `ios_base::openmode __mode`)
- locale `pubimbue` (const locale `&__loc`)

- `int_type sbumpc ()`
- `int_type sgetc ()`
- `streamsize sgetn (char_type *__s, streamsize __n)`
- `int_type snextc ()`
- `int_type sputbackc (char_type __c)`
- `int_type sputc (char_type __c)`
- `streamsize sputn (const char_type *__s, streamsize __n)`
- `int_type sungetc ()`
- `void swap (basic_filebuf &)`
- `basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)`
- `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `int pubsync ()`

### Protected Member Functions

- `void __safe_gbump (streamsize __n)`
- `void __safe_pbump (streamsize __n)`
- `void _M_allocate_internal_buffer ()`
- `bool _M_convert_to_external (char_type *, streamsize)`
- `void _M_create_pback ()`
- `void _M_destroy_internal_buffer () throw ()`
- `void _M_destroy_pback () throw ()`
- `int _M_get_ext_pos (__state_type &__state)`
- `pos_type _M_seek (off_type __off, ios_base::seekdir __way, __state_type __state)`
- `void _M_set_buffer (streamsize __off)`
- `bool _M_terminate_output ()`
- `void gbump (int __n)`
- `virtual void imbue (const locale &__loc)`
- `virtual int_type overflow (int_type __c=encoding_char_traits<_CharT>::eof())`
- `virtual int_type pbackfail (int_type __c=encoding_char_traits<_CharT>::eof())`
- `void pbump (int __n)`
- `virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)`
- `void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)`
- `void setp (char_type *__pbeg, char_type *__pend)`
- `virtual streamsize showmanyc ()`
- `void swap (basic_streambuf &__sb)`
- `virtual int sync ()`
- `virtual int_type uflow ()`
- `virtual int_type underflow ()`
- `virtual streamsize xsgetn (char_type *__s, streamsize __n)`
- `virtual streamsize xsputn (const char_type *__s, streamsize __n)`
- `char_type * eback () const`
- `char_type * gptr () const`



- `char_type * egptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * epptr () const`

### Protected Attributes

- `char_type * \_M\_buf`
  - `bool \_M\_buf\_allocated`
  - `locale \_M\_buf\_locale`
  - `size_t \_M\_buf\_size`
  - `const \_\_codecvt\_type * \_M\_codecvt`
  - `char * \_M\_ext\_buf`
  - `streamsize \_M\_ext\_buf\_size`
  - `char * \_M\_ext\_end`
  - `const char * \_M\_ext\_next`
  - `__file_type \_M\_file`
  - `char_type * \_M\_in\_beg`
  - `char_type * \_M\_in\_cur`
  - `char_type * \_M\_in\_end`
  - `__c_lock \_M\_lock`
  - `ios_base::openmode \_M\_mode`
  - `char_type * \_M\_out\_beg`
  - `char_type * \_M\_out\_cur`
  - `char_type * \_M\_out\_end`
  - `bool \_M\_reading`
  - `__state_type \_M\_state\_beg`
  - `__state_type \_M\_state\_cur`
  - `__state_type \_M\_state\_last`
  - `bool \_M\_writing`
- 
- `char_type \_M\_pback`
  - `char_type * \_M\_pback\_cur\_save`
  - `char_type * \_M\_pback\_end\_save`
  - `bool \_M\_pback\_init`

#### 5.399.1 Detailed Description

```
template<typename _CharT>
class __gnu_cxx::enc_filebuf< _CharT >
```

class `enc_filebuf`.

#### 5.399.2 Member Function Documentation

##### `_M_create_pback()`

```
void std::basic\_filebuf< _CharT, encoding\_char\_traits< _CharT > >::_M_create_pback () [inline],
[protected], [inherited]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

**`_M_destroy_pback()`**

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_destroy_pback () throw ( )
[inline], [protected], [inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

**`_M_set_buffer()`**

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_set_buffer (
    streamsize __off) [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `egptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

**`close()`**

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::__filebuf_type * std::basic_filebuf<↔
    _CharT, encoding_char_traits<_CharT>>::close () [inherited]
```

Closes the currently associated file.

**Returns**

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

**`eback()`**

```
char_type * std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::eback () const [inline],
[protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

**`egptr()`**

```
char_type * std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::egptr () const [inline],
[protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

**epptr()**

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

**gbump()**

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::gbump (int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

**getloc()**

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

**gptr()**

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

**imbue()**

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::imbue (const locale & __loc) [protected], [virtual], [inherited]
```

Changes translations.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**`in_avail()`**

```
streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::in_avail () [inline],
[inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**`is_open()`**

```
bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::is_open () const throw ()
[inline], [nodiscard], [inherited]
```

Returns true if the external file is open.

**`open()` [1/3]**

```
_If_fs_path<_Path, __filebuf_type * > std::basic_filebuf<_CharT, encoding_char_traits<_CharT>
> >::open (
    const _Path & __s,
    ios_base::openmode __mode) [inline], [inherited]
```

Opens an external file.

**Parameters**

|                     |                                                            |
|---------------------|------------------------------------------------------------|
| <code>__s</code>    | The name of the file, as a <code>filesystem::path</code> . |
| <code>__mode</code> | The open mode flags.                                       |

**Returns**

`this` on success, NULL on failure

**`open()` [2/3]**

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::__filebuf_type * std::basic_filebuf<↔
_Chart, encoding_char_traits<_CharT>> >::open (
    const char * __s,
    ios_base::openmode __mode) [inherited]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

**Returns**

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

| ios_base Flag combination |    |     |       |     | stdio equivalent |
|---------------------------|----|-----|-------|-----|------------------|
| binary                    | in | out | trunc | app |                  |
|                           |    |     |       |     |                  |
|                           |    | +   |       |     | w                |
|                           |    | +   |       | +   | a                |
|                           |    |     |       | +   | a                |
|                           |    | +   | +     |     | w                |
|                           | +  |     |       |     | r                |
|                           | +  | +   |       |     | r+               |
|                           | +  | +   | +     |     | w+               |
|                           | +  | +   |       | +   | a+               |
|                           | +  |     |       | +   | a+               |
|                           |    |     |       |     |                  |
| +                         |    | +   |       |     | wb               |
| +                         |    | +   |       | +   | ab               |
| +                         |    |     |       | +   | ab               |
| +                         |    | +   | +     |     | wb               |
| +                         | +  |     |       |     | rb               |
| +                         | +  | +   |       |     | r+b              |
| +                         | +  | +   | +     |     | w+b              |
| +                         | +  | +   |       | +   | a+b              |
| +                         | +  |     |       | +   | a+b              |

**open() [3/3]**

```
__filebuf_type * std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::open (
    const std::string & __s,
    ios_base::openmode __mode) [inline], [inherited]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

**Returns**

`this` on success, NULL on failure

**overflow()**

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type std::basic_filebuf<_CharT,
encoding_char_traits<_CharT>>::overflow (
    int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**pbackfail()**

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type std::basic_filebuf<_CharT,
encoding_char_traits<_CharT>>::pbackfail (
    int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**pbase()**

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pbase () const [inline],  
[protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

**pbump()**

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pbump (  
    int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

**pptr()**

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pptr () const [inline],  
[protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

**pubimbue()**

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubimbue (  
    const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

**pubseekoff()**

```
pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekoff` function.

**pubseekpos()**

```
pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

**pubsetbuf()**

```
basic_streambuf * std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsetbuf (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

**pubsync()**

```
int std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsync () [inline], [inherited]
```

Calls virtual `sync` function.

**sbumpc()**

```
int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sbumpc () [inline],
[inherited]
```

Getting the next character.

**Returns**

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.



**seekoff()**

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::pos_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::seekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >`.

**seekpos()**

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::pos_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::seekpos (
    pos_type __pos,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >`.

**setbuf()**

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__streambuf_type * std::basic_filebuf<
_CharT, encoding_char_traits< _CharT > >::setbuf (
    char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Manipulates the buffer.

**Parameters**

|                  |                            |
|------------------|----------------------------|
| <code>__s</code> | Pointer to a buffer area.  |
| <code>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >`.

**setg()**

```
void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

**setp()**

```
void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setp (
    char_type * __pbeg,
    char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

**sgetc()**

```
int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetc () [inline],
[inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

**sgetn()**

```
streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetn (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

|       |                |
|-------|----------------|
| $\_s$ | A buffer area. |
| $\_n$ | A count.       |

Returns `xsgetrn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

**showmanyc()**

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::showmanyc () [protected],  
[virtual], [inherited]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >`.

**snextc()**

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::snextc () [inline],  
[inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

**sputbackc()**

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputbackc (  
    char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|       |                             |
|-------|-----------------------------|
| $\_c$ | The character to push back. |
|-------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

**sputc()**

```
int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputc (
    char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↵__c)`.

**sputn()**

```
streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputn (
    const char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xputn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**sungetc()**

```
int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sungetc () [inline],
[inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

**sync()**

```
int std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::sync () [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >`.

**uflow()**

```
virtual int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::uflow () [inline], [protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

**underflow()**

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::underflow () [protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >`.

**xsggetn()**

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::xsggetn (
    char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

## Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**xsputn()**

```
streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::xsputn (
    const char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.399.3 Member Data Documentation****`_M_buf`**

```
char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf [protected],
[inherited]
```

Pointer to the beginning of internal buffer.

**`_M_buf_locale`**

```
locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_locale [protected],
[inherited]
```

Current locale setting.

### **\_M\_buf\_size**

```
size_t std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_buf_size [protected],  
[inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

### **\_M\_ext\_buf**

```
char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf [protected],  
[inherited]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

### **\_M\_ext\_buf\_size**

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf_size [protected],  
[inherited]
```

Size of buffer held by `_M_ext_buf`.

### **\_M\_ext\_next**

```
const char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_next [protected],  
[inherited]
```

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

### **\_M\_in\_beg**

```
char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_beg [protected],  
[inherited]
```

Start of get area.

### **\_M\_in\_cur**

```
char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_cur [protected],  
[inherited]
```

Current read area.

### **\_M\_in\_end**

```
char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_end [protected],  
[inherited]
```

End of get area.

### **\_M\_mode**

```
ios_base::openmode std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_mode [protected],  
[inherited]
```

Place to stash in || out || in | out settings for current filebuf.

### **\_M\_out\_beg**

```
char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_beg [protected],  
[inherited]
```

Start of put area.

**`_M_out_cur`**

```
char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_cur [protected],  
[inherited]
```

Current put area.

**`_M_out_end`**

```
char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_end [protected],  
[inherited]
```

End of put area.

**`_M_pback`**

```
char_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback [protected],  
[inherited]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

**`_M_pback_cur_save`**

```
char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_cur_save [protected],  
[inherited]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

**`_M_pback_end_save`**

```
char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_end_save [protected],  
[inherited]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

**`_M_pback_init`**

```
bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_init [protected],  
[inherited]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.



## **\_M\_reading**

```
bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_reading [protected],
[inherited]
```

\_M\_reading == false && \_M\_writing == false for **uncommitted** mode; \_M\_reading == true for **read** mode; \_M\_writing == true for **write** mode;

NB: \_M\_reading == true && \_M\_writing == true is unused.

The documentation for this class was generated from the following file:

- [enc\\_filebuf.h](#)

## **5.400 \_\_gnu\_cxx::encoding\_char\_traits< \_CharT > Struct Template Reference**

```
#include <codecvt_specializations.h>
```

Inheritance diagram for \_\_gnu\_cxx::encoding\_char\_traits< \_CharT >:



### **Public Types**

- typedef `_CharT` **char\_type**
- using **comparison\_category**
- typedef `_Char_types< _CharT >::int_type` **int\_type**
- typedef `_Char_types< _CharT >::off_type` **off\_type**
- typedef `std::fpos< state_type >` **pos\_type**
- typedef `encoding_state` **state\_type**

### **Static Public Member Functions**

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)

- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 5.400.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::encoding_char_traits< _CharT >
```

encoding\_char\_traits

The documentation for this struct was generated from the following file:

- [codecvt\\_specializations.h](#)

### 5.401 `__gnu_cxx::encoding_state` Class Reference

```
#include <codecvt_specializations.h>
```

#### Public Types

- typedef iconv\_t **descriptor\_type**

#### Public Member Functions

- **encoding\_state** (const char \* \_\_int, const char \* \_\_ext, int \_\_ibom=0, int \_\_ebom=0, int \_\_bytes=1)
- **encoding\_state** (const [encoding\\_state](#) & \_\_obj)
- int **character\_ratio** () const
- int **external\_bom** () const
- const [std::string](#) **external\_encoding** () const
- bool **good** () const throw ()
- const descriptor\_type & **in\_descriptor** () const
- int **internal\_bom** () const
- const [std::string](#) **internal\_encoding** () const
- [encoding\\_state](#) & **operator=** (const [encoding\\_state](#) & \_\_obj)
- const descriptor\_type & **out\_descriptor** () const

#### Protected Member Functions

- void **construct** (const [encoding\\_state](#) & \_\_obj)
- void **destroy** () throw ()
- void **init** ()

### Protected Attributes

- `int _M_bytes`
- `int _M_ext_bom`
- `std::string _M_ext_enc`
- `descriptor_type _M_in_desc`
- `int _M_int_bom`
- `std::string _M_int_enc`
- `descriptor_type _M_out_desc`

#### 5.401.1 Detailed Description

Extension to use `iconv` for dealing with character encodings.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

### 5.402 `std::encoding_state` Class Reference

```
#include <codecvt_specializations.h>
```

#### Public Types

- `typedef iconv_t descriptor_type`

#### Public Member Functions

- `encoding_state` (`const char *__int`, `const char *__ext`, `int __ibom=0`, `int __ebom=0`, `int __bytes=1`)
- `encoding_state` (`const encoding\_state &__obj`)
- `int character_ratio () const`
- `int external_bom () const`
- `const std::string external_encoding () const`
- `bool good () const throw ()`
- `const descriptor_type & in_descriptor () const`
- `int internal_bom () const`
- `const std::string internal_encoding () const`
- `encoding\_state & operator= (const encoding\_state &__obj)`
- `const descriptor_type & out_descriptor () const`

#### Protected Member Functions

- `void construct` (`const encoding\_state &__obj`)
- `void destroy () throw ()`
- `void init ()`

### Protected Attributes

- `int _M_bytes`
- `int _M_ext_bom`
- `std::string _M_ext_enc`
- `descriptor_type _M_in_desc`
- `int _M_int_bom`
- `std::string _M_int_enc`
- `descriptor_type _M_out_desc`

#### 5.402.1 Detailed Description

Extension to use `iconv` for dealing with character encodings.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

### 5.403 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>` Struct Template Reference

#### 5.403.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>
```

Entry compare, primary template.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

### 5.404 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>` Struct Template Reference

```
#include <entry_cmp.hpp>
```

#### Classes

- struct [type](#)

#### Public Types

- typedef `__rebind_v::const_pointer` **entry**

#### 5.404.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>
```

Specialization, false.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

### 5.405 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true>` Struct Template Reference

```
#include <entry_cmp.hpp>
```

#### Public Types

- typedef `Cmp_Fn` [type](#)

#### 5.405.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true>
```

Specialization, true.

### 5.405.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef Cmp_Fn __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >::type
Compare.
```

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

### 5.406 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, No\_Throw > Struct Template Reference

#### 5.406.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw >
```

Entry predicate primary class template.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

### 5.407 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, false > Struct Template Reference

```
#include <entry_pred.hpp>
```

#### Public Types

- typedef \_\_rebind\_v::const\_pointer **entry**

#### 5.407.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >
```

Specialization, false.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

### 5.408 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, true > Struct Template Reference

```
#include <entry_pred.hpp>
```

#### Public Types

- typedef Pred **type**

#### 5.408.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >
```

Specialization, true.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

## 5.409 `__gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >` Struct Template Reference

```
#include <eq_by_less.hpp>
```

### Public Member Functions

- `bool operator() (const Key &r_lhs, const Key &r_rhs) const`

#### 5.409.1 Detailed Description

```
template<typename Key, class Cmp_Fn>  
struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >
```

Equivalence function.

The documentation for this struct was generated from the following file:

- [eq\\_by\\_less.hpp](#)

## 5.410 `__gnu_parallel::equal_split_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::equal_split_tag`:



#### 5.410.1 Detailed Description

Selects the equal splitting variant for `std::find()`.

See also

`_GLIBCXX_FIND_EQUAL_SPLIT`

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.411 `__gnu_cxx::equal_to<_Tp>` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `__gnu_cxx::equal_to<_Tp>`:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- constexpr `bool` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

#### 5.411.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::equal_to<_Tp>
```

One of the [comparison functors](#).

#### 5.411.2 Member Typedef Documentation

##### `first_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### `result_type`

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
result_type is the return type
```

##### `second_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:
```

- [stl\\_function.h](#)

## 5.412 std::equal\_to< \_Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::equal\_to< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 5.412.1 Detailed Description

```
template<typename _Tp>
struct std::equal_to< _Tp >
```

One of the [comparison functors](#).

#### 5.412.2 Member Typedef Documentation

##### first\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### result\_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type
```



### second\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.413 std::ranges::equal\_to Struct Reference

```
#include <ranges_cmp.h>
```

### Public Types

- using `is_transparent`

### Public Member Functions

- `template<typename _Tp, typename _Up>`  
requires `equality_comparable_with<_Tp, _Up>`  
`constexpr bool operator() (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval<_Tp >()==std::declval<_Up >()))`

#### 5.413.1 Detailed Description

ranges::equal\_to function object type.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 5.414 std::equal\_to< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::equal_to< void >`:



### Public Types

- `typedef void` [first\\_argument\\_type](#)
- `typedef void` [first\\_argument\\_type](#)

- typedef `__is_transparent` `is_transparent`
- typedef bool `result_type`
- typedef bool `result_type`
- typedef void `second_argument_type`
- typedef void `second_argument_type`

### Public Member Functions

- template<typename `_Tp`, typename `_Up`>  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`< `_Tp` >(`_t`←`_t`)==`std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`)==`std::forward`< `_Up` >(`_u`))
- constexpr bool **operator()** (const void &`_x`, const void &`_y`) const
- constexpr bool **operator()** (const void &`_x`, const void &`_y`) const
- constexpr bool **operator()** (const void &`_x`, const void &`_y`) const

#### 5.414.1 Detailed Description

One of the [comparison functions](#).

#### 5.414.2 Member Typedef Documentation

##### `first_argument_type` [1/2]

typedef void `std::binary_function`< void, void, bool >::`first_argument_type`  
`first_argument_type` is the type of the first argument

##### `first_argument_type` [2/2]

typedef void `std::binary_function`< void, void, bool >::`first_argument_type`  
`first_argument_type` is the type of the first argument

##### `result_type` [1/2]

typedef bool `std::binary_function`< void, void, bool >::`result_type`  
`result_type` is the return type

##### `result_type` [2/2]

typedef bool `std::binary_function`< void, void, bool >::`result_type`  
`result_type` is the return type

##### `second_argument_type` [1/2]

typedef void `std::binary_function`< void, void, bool >::`second_argument_type`  
`second_argument_type` is the type of the second argument

##### `second_argument_type` [2/2]

typedef void `std::binary_function`< void, void, bool >::`second_argument_type`  
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.415 std::error\_category Class Reference

```
#include <system_error>
```

### Public Member Functions

- **error\_category** (const [error\\_category](#) &)=delete
- virtual [error\\_condition](#) **default\_error\_condition** (int \_\_i) const noexcept
- virtual bool [equivalent](#) (const [error\\_code](#) &\_\_code, int \_\_i) const noexcept
- virtual bool [equivalent](#) (int \_\_i, const [error\\_condition](#) &\_\_cond) const noexcept
- virtual [string](#) **message** (int) const =0
- virtual const char \* **name** () const noexcept=0
- strong\_ordering **operator<=>** (const [error\\_category](#) &\_\_rhs) const noexcept
- [error\\_category](#) & **operator=** (const [error\\_category](#) &)=delete
- bool **operator==** (const [error\\_category](#) &\_\_other) const noexcept

### 5.415.1 Detailed Description

Abstract base class for types defining a category of error codes.

An error category defines a context that gives meaning to the integer stored in an [error\\_code](#) or [error\\_condition](#) object. For example, the standard `errno` constants such as `EINVAL` and `ENOMEM` are associated with the "generic" category and other OS-specific error numbers are associated with the "system" category, but a user-defined category might give different meanings to the same numerical values.

A user-defined category can override the `equivalent` member functions to define correspondence between errors in different categories. For example, a category for errors from disk I/O could consider some of its error numbers equivalent to `ENOSPC` and `ENOENT` in the generic category.

Since

C++11

### 5.415.2 Member Function Documentation

#### **default\_error\_condition()**

```
virtual error\_condition std::error_category::default_error_condition (
    int __i) const [virtual], [noexcept]
```

Return an [error\\_condition](#) corresponding to `i` in this category.

#### **equivalent()** [1/2]

```
virtual bool std::error_category::equivalent (
    const error\_code & __code,
    int __i) const [virtual], [noexcept]
```

Test whether `code` corresponds to `i` for this category.

#### **equivalent()** [2/2]

```
virtual bool std::error_category::equivalent (
    int __i,
    const error\_condition & __cond) const [virtual], [noexcept]
```

Test whether `cond` corresponds to `i` for this category.

**message()**

```
virtual string std::error_category::message (
    int ) const [pure virtual]
```

A description of the error condition corresponding to the number.

**name()**

```
virtual const char * std::error_category::name () const [pure virtual], [noexcept]
```

A string that identifies the error category.

**operator<=>()**

```
strong_ordering std::error_category::operator<=> (
    const error_category & __rhs) const [inline], [nodiscard], [noexcept]
```

Ordered comparison that defines a total order for error categories.

**operator==()**

```
bool std::error_category::operator== (
    const error_category & __other) const [inline], [noexcept]
```

An error\_category only compares equal to itself.

The documentation for this class was generated from the following file:

- [system\\_error](#)

**5.416 std::error\_code Class Reference**

```
#include <system_error>
```

**Public Member Functions**

- `template<typename _ErrorCodeEnum, typename = _Check<_ErrorCodeEnum>>`  
`error_code` (`_ErrorCodeEnum __e`) `noexcept`
- `error_code` (`const error_code &`)=`default`
- `error_code` (`int __v, const error_category &__cat`) `noexcept`
- `void assign` (`int __v, const error_category &__cat`) `noexcept`
- `const error_category & category` () `const noexcept`
- `void clear` () `noexcept`
- `error_condition default_error_condition` () `const noexcept`
- `string message` () `const`
- `operator bool` () `const noexcept`
- `error_code & operator=` (`const error_code &`)=`default`
- `int value` () `const noexcept`

**Related Symbols**

(Note that these are not member symbols.)

- `error_code make_error_code` (`errc __e`) `noexcept`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream`< `_CharT, _Traits` > & `operator<<` (`basic_ostream`< `_CharT, _Traits` > & `__os, const error_code` & `__e`)
- `strong_ordering operator<=>` (`const error_code &__lhs, const error_code &__rhs`) `noexcept`

### 5.416.1 Detailed Description

Class `error_code`

This class is a value type storing an integer error number and a category that gives meaning to the error number. Typically this is done close to the point where the error happens, to capture the original error value.

An `error_code` object can be used to store the original error value emitted by some subsystem, with a category relevant to the subsystem. For example, errors from POSIX library functions can be represented by an `errno` value and the "generic" category, but errors from an HTTP library might be represented by an HTTP response status code (e.g. 404) and a custom category defined by the library.

Since

C++11

### 5.416.2 Constructor & Destructor Documentation

#### `error_code()`

```
template<typename _ErrorCodeEnum, typename = _Check<_ErrorCodeEnum>>
std::error_code::error_code (
    _ErrorCodeEnum __e) [inline], [noexcept]
```

Initialize with a user-defined type, by calling `make_error_code`.

### 5.416.3 Member Function Documentation

#### `category()`

```
const error\_category & std::error_code::category () const [inline], [noexcept]
```

The error category that this error belongs to.

#### `default_error_condition()`

```
error\_condition std::error_code::default_error_condition () const [noexcept]
```

An `error_condition` for this error's category and value.

#### `message()`

```
string std::error_code::message () const [inline]
```

The category's description of the value.

#### `operator bool()`

```
std::error_code::operator bool () const [inline], [explicit], [noexcept]
```

Test whether `value()` is non-zero.

#### `value()`

```
int std::error_code::value () const [inline], [noexcept]
```

The error value.

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 5.417 `std::error_condition` Class Reference

```
#include <system_error>
```

**Public Member Functions**

- `error_condition ()` noexcept
- `template<typename _ErrorConditionEnum, typename = _Check<_ErrorConditionEnum>>> error_condition (_ErrorConditionEnum __e)` noexcept
- `error_condition (const error_condition &)=default`
- `error_condition (int __v, const error_category &__cat)` noexcept
- `void assign (int __v, const error_category &__cat)` noexcept
- `const error_category & category ()` const noexcept
- `void clear ()` noexcept
- `string message ()` const
- `operator bool ()` const noexcept
- `error_condition & operator= (const error_condition &)=default`
- `int value ()` const noexcept

**Related Symbols**

(Note that these are not member symbols.)

- `error_condition make_error_condition (errc __e)` noexcept
- `strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs)` noexcept
- `bool operator== (const error_code &__lhs, const error_code &__rhs)` noexcept
- `bool operator== (const error_code &__lhs, const error_condition &__rhs)` noexcept
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs)` noexcept

**5.417.1 Detailed Description**

Class `error_condition`

This class represents error conditions that may be visible at an API boundary. Different `error_code` values that can occur within a library or module might map to the same `error_condition`.

An `error_condition` represents something that the program can test for, and subsequently take appropriate action.

Since

C++11

**5.417.2 Constructor & Destructor Documentation****error\_condition() [1/3]**

```
std::error_condition::error_condition () [inline], [noexcept]
```

Initialize with a zero (no error) value and the generic category.

**error\_condition() [2/3]**

```
std::error_condition::error_condition (
    int __v,
    const error_category & __cat) [inline], [noexcept]
```

Initialize with the specified value and category.

**error\_condition() [3/3]**

```
template<typename _ErrorConditionEnum, typename = _Check<_ErrorConditionEnum>>
std::error_condition::error_condition (
    _ErrorConditionEnum __e) [inline], [noexcept]
```

Initialize with a user-defined type, by calling `make_error_condition`.

### 5.417.3 Member Function Documentation

#### **assign()**

```
void std::error_condition::assign (
    int __v,
    const error\_category & __cat) [inline], [noexcept]
```

Set the value and category.

#### **category()**

```
const error\_category & std::error_condition::category () const [inline], [noexcept]
```

The error category that this error belongs to.

#### **clear()**

```
void std::error_condition::clear () [inline], [noexcept]
```

Reset the value and category to the default-constructed state.

#### **message()**

```
string std::error_condition::message () const [inline]
```

The category's description of the value.

#### **operator bool()**

```
std::error_condition::operator bool () const [inline], [explicit], [noexcept]
```

Test whether `value()` is non-zero.

#### **value()**

```
int std::error_condition::value () const [inline], [noexcept]
```

The error value.

The documentation for this class was generated from the following file:

- [system\\_error](#)

### 5.418 `__gnu_parallel::exact_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::exact_tag`:



**Public Member Functions**

- **exact\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

**5.418.1 Detailed Description**

Forces parallel merging with exact splitting, at compile time.

**5.418.2 Member Function Documentation****\_\_get\_num\_threads()**

[\\_ThreadIndex](#) [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads](#) () [inline], [inherited]

Find out desired number of threads.

**Returns**

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

**set\_num\_threads()**

void [\\_\\_gnu\\_parallel::parallel\\_tag::set\\_num\\_threads](#) (  
[\\_ThreadIndex](#) \_\_num\_threads) [inline], [inherited]

Set the desired number of threads.

**Parameters**

|                               |                            |
|-------------------------------|----------------------------|
| <a href="#">__num_threads</a> | Desired number of threads. |
|-------------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

**5.419 std::exception Class Reference**

```
#include <exception.h>
```



Inheritance diagram for `std::exception`:



## Public Member Functions

- `_GLIBCXX26_CONSTEXPR exception (const exception &)=default`
- `_GLIBCXX26_CONSTEXPR exception (exception &&)=default`
- `_GLIBCXX26_CONSTEXPR exception & operator= (const exception &)=default`
- `_GLIBCXX26_CONSTEXPR exception & operator= (exception &&)=default`
- `virtual const char * what () const noexcept`

### 5.419.1 Detailed Description

Base class for all library exceptions.

This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

### 5.419.2 Member Function Documentation

#### `what()`

```
virtual const char * std::exception::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented in `std::bad_alloc`, `std::bad_cast`, `std::bad_exception`, `std::bad_function_call`, `std::bad_typeid`, `std::bad_weak_ptr`, `std::experimental::filesystem::v1::filesystem_error`, `std::experimental::fundamentals_v1::bad_any_cast`, `std::filesystem::filesystem_error`, `std::future_error`, `std::logic_error`, and `std::runtime_error`.

The documentation for this class was generated from the following file:

- [exception.h](#)

## 5.420 `std::__unspecified__::exception_ptr` Class Reference

```
#include <exception>
```

### Public Member Functions

- `exception_ptr` (const [exception\\_ptr](#) &) noexcept
- `exception_ptr` ([exception\\_ptr](#) &&\_\_o) noexcept
- `exception_ptr` (nullptr\_t) noexcept
- const class `std::type_info * __cxa_exception_type` () const noexcept
- `operator bool` () const noexcept
- `exception_ptr` & `operator=` (const [exception\\_ptr](#) &) noexcept
- `exception_ptr` & `operator=` ([exception\\_ptr](#) &&\_\_o) noexcept
- void `swap` ([exception\\_ptr](#) &) noexcept

### Friends

- bool `operator==` (const [exception\\_ptr](#) &, const [exception\\_ptr](#) &) noexcept=default
- `exception_ptr` `std::current_exception` () noexcept
- template<typename \_Ex>  
[exception\\_ptr](#) `std::make_exception_ptr` (\_Ex) noexcept
- void `std::rethrow_exception` ([exception\\_ptr](#))

### Related Symbols

(Note that these are not member symbols.)

- void `swap` ([exception\\_ptr](#) &\_\_lhs, [exception\\_ptr](#) &\_\_rhs)

### 5.420.1 Detailed Description

An opaque pointer to an arbitrary exception.

The actual name of this type is unspecified, so the alias `std::exception_ptr` should be used to refer to it.

Since

C++11 (but usable in C++98 as a GCC extension)

The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

## 5.421 `std::exception_ptr` Class Reference

```
#include <exception>
```

### Public Member Functions

- **exception\_ptr** (const [exception\\_ptr](#) &) noexcept
- **exception\_ptr** ([exception\\_ptr](#) &&\_\_o) noexcept
- **exception\_ptr** (nullptr\_t) noexcept
- const class [std::type\\_info](#) \* **\_\_cxa\_exception\_type** () const noexcept
- **operator bool** () const noexcept
- [exception\\_ptr](#) & **operator=** (const [exception\\_ptr](#) &) noexcept
- [exception\\_ptr](#) & **operator=** ([exception\\_ptr](#) &&\_\_o) noexcept
- void **swap** ([exception\\_ptr](#) &) noexcept

### Friends

- bool **operator==** (const [exception\\_ptr](#) &, const [exception\\_ptr](#) &) noexcept=default
- [exception\\_ptr](#) **std::current\_exception** () noexcept
- template<typename \_Ex>  
[exception\\_ptr](#) **std::make\_exception\_ptr** (\_Ex) noexcept
- void **std::rethrow\_exception** ([exception\\_ptr](#))

### Related Symbols

(Note that these are not member symbols.)

- void **swap** ([exception\\_ptr](#) &\_\_lhs, [exception\\_ptr](#) &\_\_rhs)

#### 5.421.1 Detailed Description

An opaque pointer to an arbitrary exception.

The actual name of this type is unspecified, so the alias `std::exception_ptr` should be used to refer to it.

Since

C++11 (but usable in C++98 as a GCC extension)

The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

#### 5.422 `std::exponential_distribution<_RealType>` Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- [exponential\\_distribution](#) ()
- [exponential\\_distribution](#) (`_RealType` \_\_lambda)
- [exponential\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)

- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const **param\_type** &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** (**result\_type** \*\_\_f, **result\_type** \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const **param\_type** &\_\_p)
- \_RealType **lambda** () const
- **result\_type** **max** () const
- **result\_type** **min** () const
- template<typename \_UniformRandomNumberGenerator>  
**result\_type** **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
**result\_type** **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const **param\_type** &\_\_p)
- **param\_type** **param** () const
- void **param** (const **param\_type** &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const **exponential\_distribution** &\_\_d1, const **exponential\_distribution** &\_\_d2)

### 5.422.1 Detailed Description

template<typename \_RealType = double>  
class std::exponential\_distribution<\_RealType>

An exponential continuous distribution for random numbers.

The formula for the exponential probability density function is  $p(x|\lambda) = \lambda e^{-\lambda x}$ .

**Table 2344 Distribution Statistics**

|                    |                         |
|--------------------|-------------------------|
| Mean               | $\frac{1}{\lambda}$     |
| Median             | $\frac{\ln 2}{\lambda}$ |
| Mode               | <i>zero</i>             |
| Range              | $[0, \infty]$           |
| Standard Deviation | $\frac{1}{\lambda}$     |

Since

C++11

### 5.422.2 Member Typedef Documentation

#### result\_type

template<typename \_RealType = double>  
typedef \_RealType std::exponential\_distribution<\_RealType>::result\_type  
The type of the range of the distribution.

### 5.422.3 Constructor & Destructor Documentation

#### exponential\_distribution() [1/2]

template<typename \_RealType = double>  
std::exponential\_distribution<\_RealType>::exponential\_distribution () [inline]  
Constructs an exponential distribution with inverse scale parameter 1.0.

**exponential\_distribution()** [2/2]

```
template<typename _RealType = double>
std::exponential_distribution< _RealType >::exponential_distribution (
    _RealType __lambda) [inline], [explicit]
```

Constructs an exponential distribution with inverse scale parameter  $\lambda$ .

**5.422.4 Member Function Documentation****lambda()**

```
template<typename _RealType = double>
_RealType std::exponential_distribution< _RealType >::lambda () const [inline]
```

Returns the inverse scale parameter of the distribution.

**max()**

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

**min()**

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

**operator>()**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::exponential_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

**param()** [1/2]

```
template<typename _RealType = double>
param_type std::exponential_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

**param()** [2/2]

```
template<typename _RealType = double>
void std::exponential_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::exponential_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Has no effect on exponential distributions.

**5.422.5 Friends And Related Symbol Documentation****operator==**

```
template<typename _RealType = double>
bool operator== (
    const exponential_distribution< _RealType > & __d1,
    const exponential_distribution< _RealType > & __d2) [friend]
```

Return true if two exponential distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.423 std::extent< typename, \_UInt > Struct Template Reference**

```
#include <type_traits>
```

Inheritance diagram for std::extent< typename, \_UInt >:

**Public Types**

- using **type**
- using **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** () const noexcept

**Static Public Attributes**

- static constexpr **size\_t value**

### 5.423.1 Detailed Description

```
template<typename, unsigned _Uint = 0>
struct std::extent< typename, _Uint >
```

extent

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.424 std::extreme\_value\_distribution< \_RealType > Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **extreme\_value\_distribution** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- **extreme\_value\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

### Friends

- bool **operator==** (const [extreme\\_value\\_distribution](#) &\_\_d1, const [extreme\\_value\\_distribution](#) &\_\_d2)

### 5.424.1 Detailed Description

```
template<typename _RealType = double>
class std::extreme_value_distribution< _RealType >
```

A extreme\_value\_distribution random number distribution.  
The formula for the normal probability mass function is

$$p(x|a,b) = \frac{1}{b} \exp\left(\frac{a-x}{b} - \exp\left(\frac{a-x}{b}\right)\right)$$

Since

C++11

### 5.424.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::extreme_value_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 5.424.3 Member Function Documentation

#### a()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution< _RealType >::a () const [inline]
```

Return the  $a$  parameter of the distribution.

#### b()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution< _RealType >::b () const [inline]
```

Return the  $b$  parameter of the distribution.

#### max()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::extreme_value_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).



**param()** [1/2]

```
template<typename _RealType = double>
param_type std::extreme_value_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.  
Referenced by [std::operator>>\(\)](#).

**param()** [2/2]

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

**5.424.4 Friends And Related Symbol Documentation****operator==**

```
template<typename _RealType = double>
bool operator== (
    const extreme_value_distribution< _RealType > & __d1,
    const extreme_value_distribution< _RealType > & __d2) [friend]
```

Return true if two extreme value distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.425 std::locale::facet Class Reference**

```
#include <locale_classes.h>
```

Inheritance diagram for std::locale::facet:



### Protected Member Functions

- **facet** (const [facet](#) &)=delete
- **facet** (size\_t \_\_refs=0) throw ()

- virtual `~facet()`
- `facet` & `operator=` (const `facet` &)=delete

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char * __s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char * __s)`

### Friends

- class `locale`
- class `locale::_Impl`

### 5.425.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Facets may not be copied or assigned.

### 5.425.2 Constructor & Destructor Documentation

#### `facet()`

```
std::locale::facet::facet (
    size_t __refs = 0) throw ( )    [inline], [explicit], [protected]
```

Facet constructor.

This is the constructor provided by the standard. If refs is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__refs</code> | The initial value for reference count. |
|---------------------|----------------------------------------|

Referenced by `std::__cxx11::collate<_CharT>::collate()`, `std::__cxx11::collate<_CharT>::collate()`, `std::messages<_CharT>::messages()`, `std::messages<_CharT>::messages()`, `std::money_get<_CharT, _InIter>::money_get()`, `std::money_put<_CharT, _OutIter>::money_put()`, `std::moneypunct<_CharT, _Intl>::moneypunct()`, `std::moneypunct<_CharT, _Intl>::moneypunct()`, `std::moneypunct<_CharT, _Intl>::moneypunct()`, `std::num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>::num_get()`, `std::num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>::num_put()`, `std::num_punct<_CharT>::num_punct()`, `std::num_punct<_CharT>::num_punct()`, `std::num_punct<_CharT>::num_punct()`, `std::time_get<_CharT, _InIter>::time_get()`, `std::time_put<_CharT, _OutIter>::time_put()`, and `~facet()`.

#### `~facet()`

```
virtual std::locale::facet::~~facet ()    [protected], [virtual]
```

Facet destructor.

References `facet()`.

The documentation for this class was generated from the following file:

- `locale_classes.h`

## 5.426 std::filesystem::file\_status Class Reference

```
#include <filesystem>
```

### Public Member Functions

- **file\_status** (const [file\\_status](#) &) noexcept=default
- **file\_status** ([file\\_status](#) &&) noexcept=default
- **file\_status** ([file\\_type](#) \_\_ft, [perms](#) \_\_prms=perms::unknown) noexcept
- **file\_status** & **operator=** (const [file\\_status](#) &) noexcept=default
- **file\_status** & **operator=** ([file\\_status](#) &&) noexcept=default
- [perms](#) **permissions** () const noexcept
- void **permissions** ([perms](#) \_\_prms) noexcept
- [file\\_type](#) **type** () const noexcept
- void **type** ([file\\_type](#) \_\_ft) noexcept

### Friends

- bool **operator==** (const [file\\_status](#) &, const [file\\_status](#) &) noexcept=default

#### 5.426.1 Detailed Description

Information about a file's type and permissions.

Since

C++17

The documentation for this class was generated from the following file:

- [bits/fs\\_dir.h](#)

## 5.427 std::experimental::filesystem::v1::filesystem\_error Class Reference

```
#include <fs_path.h>
```

Inheritance diagram for `std::experimental::filesystem::v1::filesystem_error`:



### Public Member Functions

- **filesystem\_error** (const [string](#) & \_\_what\_arg, const [path](#) & \_\_p1, const [path](#) & \_\_p2, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) & \_\_what\_arg, const [path](#) & \_\_p1, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) & \_\_what\_arg, [error\\_code](#) \_\_ec)
- const [error\\_code](#) & **code** () const noexcept
- const [path](#) & **path1** () const noexcept
- const [path](#) & **path2** () const noexcept
- const char \* **what** () const noexcept

### 5.427.1 Detailed Description

Exception type thrown by the Filesystem TS library.

### 5.427.2 Member Function Documentation

#### what()

```
const char * std::experimental::filesystem::v1::filesystem_error::what () const [inline], [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor). Reimplemented from [std::runtime\\_error](#).

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)

## 5.428 std::filesystem::filesystem\_error Class Reference

```
#include <filesystem>
```

Inheritance diagram for std::filesystem::filesystem\_error:



### Public Member Functions

- **filesystem\_error** (const [filesystem\\_error](#) &)=default
- **filesystem\_error** (const [string](#) &\_\_what\_arg, const [path](#) &\_\_p1, const [path](#) &\_\_p2, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) &\_\_what\_arg, const [path](#) &\_\_p1, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) &\_\_what\_arg, [error\\_code](#) \_\_ec)
- const [error\\_code](#) & **code** () const noexcept
- **filesystem\_error** & **operator=** (const [filesystem\\_error](#) &)=default
- const [path](#) & **path1** () const noexcept
- const [path](#) & **path2** () const noexcept
- const char \* **what** () const noexcept

### 5.428.1 Detailed Description

Exception type thrown by the Filesystem library.

Since

C++17

### 5.428.2 Member Function Documentation

#### what()

```
const char * std::filesystem::filesystem_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::runtime\\_error](#).

References [std::make\\_error\\_code\(\)](#).

The documentation for this class was generated from the following file:

- [bits/fs\\_path.h](#)

## 5.429 `__gnu_parallel::find_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::find_tag`:



### 5.429.1 Detailed Description

Base class for `std::find()` variants.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.430 `std::fisher_f_distribution<_RealType>` Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- **fisher\_f\_distribution** (`_RealType __m, _RealType __n=_RealType(1)`)
- **fisher\_f\_distribution** (const [param\\_type](#) &\_\_p)

- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **\_\_generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
void **\_\_generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [\\_RealType](#) **m** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- [\\_RealType](#) **n** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator**() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator**() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::fisher\\_f\\_distribution](#)< \_RealType1 > &\_\_x)
- bool **operator==** (const [fisher\\_f\\_distribution](#) &\_\_d1, const [fisher\\_f\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::fisher\\_f\\_distribution](#)< \_RealType1 > &\_\_x)

## 5.430.1 Detailed Description

**template<typename \_RealType = double>**  
**class std::fisher\_f\_distribution<\_RealType>**

A fisher\_f\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Since

C++11

## 5.430.2 Member Typedef Documentation

### result\_type

template<typename \_RealType = double>  
typedef \_RealType [std::fisher\\_f\\_distribution](#)< \_RealType >::result\_type

The type of the range of the distribution.



### 5.430.3 Member Function Documentation

#### max()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::fisher_f_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

#### param() [1/2]

```
template<typename _RealType = double>
param_type std::fisher_f_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### param() [2/2]

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

#### reset()

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

### 5.430.4 Friends And Related Symbol Documentation

#### operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::fisher_f_distribution< _RealType1 > & __x) [friend]
```

Inserts a fisher\_f\_distribution random number distribution \_\_x into the output stream \_\_os.

## Parameters

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                |
| <code>__x</code>  | A <code>fisher_f_distribution</code> random number distribution. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
    const fisher\_f\_distribution< _RealType > & __d1,
    const fisher\_f\_distribution< _RealType > & __d2) [friend]
```

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic\_istream< _CharT, _Traits > & operator>> (
    std::basic\_istream< _CharT, _Traits > & __is,
    std::fisher\_f\_distribution< _RealType1 > & __x) [friend]
```

Extracts a `fisher_f_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                     |
| <code>__x</code>  | A <code>fisher_f_distribution</code> random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.431 `__gnu_cxx::forced_error` Struct Reference**

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::forced_error`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 5.431.1 Detailed Description

Thrown by utilities for testing exception safety.

#### 5.431.2 Member Function Documentation

##### **what()**

```
virtual const char * std::exception::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad\\_alloc](#), [std::bad\\_cast](#), [std::bad\\_exception](#), [std::bad\\_function\\_call](#), [std::bad\\_typeid](#), [std::bad\\_weak\\_ptr](#), [std::experimental::filesystem::v1::filesystem\\_error](#), [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#), [std::filesystem::filesystem\\_error](#), [std::future\\_error](#), [std::logic\\_error](#), and [std::runtime\\_error](#).

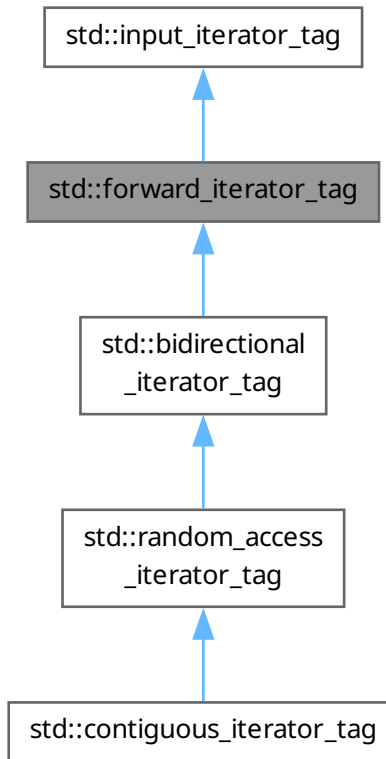
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.432 `std::forward_iterator_tag` Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for std::forward\_iterator\_tag:



#### 5.432.1 Detailed Description

Forward iterators support a superset of input iterator operations.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 5.433 std::\_\_debug::forward\_list< \_Tp, \_Alloc > Class Template Reference

```
#include <forward_list>
```

Inheritance diagram for `std::__debug::forward_list<_Tp, _Alloc>`:



## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator<_Base_const_iterator, forward_list>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator<_Base_iterator, forward_list>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **forward\_list** (`_Base_ref __x`)
- template<typename `_InputIterator`, typename = `std::RequireInputIter<_InputIterator>`>>  
**forward\_list** (`_InputIterator __first, _InputIterator __last, const allocator_type &__al=allocator_type()`)
- **forward\_list** (`const allocator_type &__al`) noexcept
- **forward\_list** (`const forward_list &`)=default
- **forward\_list** (`const forward_list &__list, const allocator_type &__al`)
- **forward\_list** (`forward_list &&`)=default
- **forward\_list** (`forward_list &&__list, const allocator_type &__al`) noexcept(`std::is_nothrow_constructible<_Base, _Base, const allocator_type &>::value`)
- **forward\_list** (`size_type __n, const __type_identity_t<_Tp> &__value, const allocator_type &__al=allocator_type()`)
- **forward\_list** (`size_type __n, const allocator_type &__al=allocator_type()`)
- **forward\_list** (`std::initializer_list<_Tp> __il, const allocator_type &__al=allocator_type()`)
- const `_Base &_M_base` () const noexcept
- `_Base &_M_base` () noexcept
- template<typename `_InputIterator`, typename = `std::RequireInputIter<_InputIterator>`>>  
void **assign** (`_InputIterator __first, _InputIterator __last`)
- void **assign** (`size_type __n, const _Tp &__val`)
- void **assign** (`std::initializer_list<_Tp> __il`)
- `const_iterator` **before\_begin** () const noexcept
- `iterator` **before\_begin** () noexcept
- `const_iterator` **begin** () const noexcept
- `iterator` **begin** () noexcept
- `const_iterator` **cbefore\_begin** () const noexcept
- `const_iterator` **cbegin** () const noexcept

- [const\\_iterator](#) **cend** () const noexcept
- void **clear** () noexcept
- template<typename... \_Args>  
[iterator](#) **emplace\_after** ([const\\_iterator](#) \_\_pos, \_Args &&... \_\_args)
- [const\\_iterator](#) **end** () const noexcept
- [iterator](#) **end** () noexcept
- [iterator](#) **erase\_after** ([const\\_iterator](#) \_\_pos)
- [iterator](#) **erase\_after** ([const\\_iterator](#) \_\_pos, [const\\_iterator](#) \_\_last)
- reference **front** ()
- const\_reference **front** () const
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>  
[iterator](#) **insert\_after** ([const\\_iterator](#) \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator](#) **insert\_after** ([const\\_iterator](#) \_\_pos, \_Tp &&\_\_val)
- [iterator](#) **insert\_after** ([const\\_iterator](#) \_\_pos, const \_Tp &\_\_val)
- [iterator](#) **insert\_after** ([const\\_iterator](#) \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- [iterator](#) **insert\_after** ([const\\_iterator](#) \_\_pos, [std::initializer\\_list](#)<\_Tp> \_\_il)
- void **merge** ([forward\\_list](#) &&\_\_list)
- template<typename \_Comp>  
void **merge** ([forward\\_list](#) &&\_\_list, \_Comp \_\_comp)
- void **merge** ([forward\\_list](#) &\_\_list)
- template<typename \_Comp>  
void **merge** ([forward\\_list](#) &\_\_list, \_Comp \_\_comp)
- [forward\\_list](#) & **operator=** (const [forward\\_list](#) &)=default
- [forward\\_list](#) & **operator=** ([forward\\_list](#) &&)=default
- [forward\\_list](#) & **operator=** ([std::initializer\\_list](#)<\_Tp> \_\_il)
- void **pop\_front** ()
- \_\_remove\_return\_type **remove** (const \_Tp &\_\_val)
- template<typename \_Pred>  
\_\_remove\_return\_type **remove\_if** (\_Pred \_\_pred)
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const value\_type &\_\_val)
- void **splice\_after** ([const\\_iterator](#) \_\_pos, [forward\\_list](#) &&\_\_list)
- void **splice\_after** ([const\\_iterator](#) \_\_pos, [forward\\_list](#) &&\_\_list, [const\\_iterator](#) \_\_before, [const\\_iterator](#) \_\_last)
- void **splice\_after** ([const\\_iterator](#) \_\_pos, [forward\\_list](#) &&\_\_list, [const\\_iterator](#) \_\_i)
- void **splice\_after** ([const\\_iterator](#) \_\_pos, [forward\\_list](#) &\_\_list)
- void **splice\_after** ([const\\_iterator](#) \_\_pos, [forward\\_list](#) &\_\_list, [const\\_iterator](#) \_\_before, [const\\_iterator](#) \_\_last)
- void **splice\_after** ([const\\_iterator](#) \_\_pos, [forward\\_list](#) &\_\_list, [const\\_iterator](#) \_\_i)
- void **swap** ([forward\\_list](#) &\_\_list) noexcept(noexcept([declval](#)<\_Base &>().swap(\_\_list)))
- \_\_remove\_return\_type **unique** ()
- template<typename \_BinPred>  
\_\_remove\_return\_type **unique** (\_BinPred \_\_binary\_pred)

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::[\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)

### 5.433.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::__debug::forward_list< _Tp, _Alloc >
```

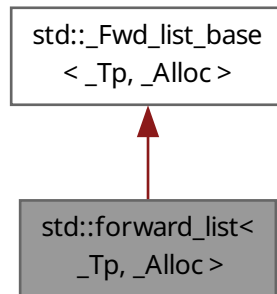
Class std::forward\_list with safety/checking/debug instrumentation.  
The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

### 5.434 std::forward\_list< \_Tp, \_Alloc > Class Template Reference

```
#include <forward_list>
```

Inheritance diagram for std::forward\_list< \_Tp, \_Alloc >:



#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `const value_type &` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `value_type &` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- [forward\\_list](#) ()=default
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
[forward\\_list](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `const _Alloc &` \_\_al=`_Alloc()`)
- [forward\\_list](#) (`const _Alloc &` \_\_al) noexcept
- [forward\\_list](#) (`const forward_list &` \_\_list)
- [forward\\_list](#) (`const forward_list &` \_\_list, `const __type_identity_t<_Alloc> &` \_\_al)
- [forward\\_list](#) ([forward\\_list](#) &&)=default

- `forward_list` (`forward_list` &&\_\_list, const \_\_type\_identity\_t<\_Alloc> &\_\_al) noexcept(`_Node_alloc_traits::S`←`_always_equal()`)
- `forward_list` (size\_type \_\_n, const \_Alloc &\_\_al= \_Alloc())
- `forward_list` (size\_type \_\_n, const \_Tp &\_\_value, const \_Alloc &\_\_al= \_Alloc())
- `forward_list` (`std::initializer_list`<\_Tp> \_\_il, const \_Alloc &\_\_al= \_Alloc())
- `~forward_list` () noexcept
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void `assign` (size\_type \_\_n, const \_Tp &\_\_val)
- void `assign` (`std::initializer_list`<\_Tp> \_\_il)
- const\_iterator `before_begin` () const noexcept
- iterator `before_begin` () noexcept
- const\_iterator `begin` () const noexcept
- iterator `begin` () noexcept
- const\_iterator `cbegin` () const noexcept
- const\_iterator `cbegin` () const noexcept
- const\_iterator `cend` () const noexcept
- void `clear` () noexcept
- template<typename... \_Args>  
iterator `emplace_after` (const\_iterator \_\_pos, \_Args &&... \_\_args)
- template<typename... \_Args>  
reference `emplace_front` (\_Args &&... \_\_args)
- bool `empty` () const noexcept
- const\_iterator `end` () const noexcept
- iterator `end` () noexcept
- iterator `erase_after` (const\_iterator \_\_pos)
- iterator `erase_after` (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference `front` ()
- const\_reference `front` () const
- allocator\_type `get_allocator` () const noexcept
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
iterator `insert_after` (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator `insert_after` (const\_iterator \_\_pos, \_Tp &&\_\_val)
- iterator `insert_after` (const\_iterator \_\_pos, const \_Tp &\_\_val)
- iterator `insert_after` (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- iterator `insert_after` (const\_iterator \_\_pos, `std::initializer_list`<\_Tp> \_\_il)
- size\_type `max_size` () const noexcept
- void `merge` (`forward_list` &&\_\_list)
- template<typename \_Comp>  
void `merge` (`forward_list` &&\_\_list, \_Comp \_\_comp)
- void `merge` (`forward_list` &\_\_list)
- template<typename \_Comp>  
void `merge` (`forward_list` &\_\_list, \_Comp \_\_comp)
- `forward_list` & `operator=` (const `forward_list` &\_\_list)
- `forward_list` & `operator=` (`forward_list` &&\_\_list) noexcept(`_Node_alloc_traits::S`←`_nothrow_move()`)
- `forward_list` & `operator=` (`std::initializer_list`<\_Tp> \_\_il)
- void `pop_front` ()
- void `push_front` (\_Tp &&\_\_val)
- void `push_front` (const \_Tp &\_\_val)
- \_\_remove\_return\_type `remove` (const \_Tp &\_\_val)



- `template<typename _Pred>`  
`__remove_return_type remove_if (_Pred __pred)`
- `template<typename _Pred>`  
`auto remove_if (_Pred __pred) -> __remove_return_type`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const value_type &__val)`
- `void reverse () noexcept`
- `void sort ()`
- `template<typename _Comp>`  
`void sort (_Comp __comp)`
- `void splice_after (const_iterator __pos, forward_list && __list) noexcept`
- `void splice_after (const_iterator __pos, forward_list && __list, const_iterator __i) noexcept`
- `void splice_after (const_iterator __pos, forward_list & __list) noexcept`
- `void splice_after (const_iterator __pos, forward_list & __list, const_iterator __i) noexcept`
- `void swap (forward_list & __list) noexcept`
- `__remove_return_type unique ()`
- `template<typename _BinPred>`  
`__remove_return_type unique (_BinPred __binary_pred)`
- `template<typename _BinPred>`  
`auto unique (_BinPred __binary_pred) -> __remove_return_type`
- `void splice_after (const_iterator __pos, forward_list &&, const_iterator __before, const_iterator __last) noexcept`
- `void splice_after (const_iterator __pos, forward_list &, const_iterator __before, const_iterator __last) noexcept`

#### 5.434.1 Detailed Description

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
class std::forward_list< _Tp, _Alloc >
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Since

C++11

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

### 5.434.2 Constructor & Destructor Documentation

#### forward\_list() [1/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list () [default]
```

Creates a forward\_list with no elements.

Referenced by [insert\\_after\(\)](#), [insert\\_after\(\)](#), [merge\(\)](#), [operator=\(\)](#), [remove\(\)](#), [resize\(\)](#), [resize\(\)](#), [sort\(\)](#), and [splice\\_after\(\)](#).

#### forward\_list() [2/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    const _Alloc & __al) [inline], [explicit], [noexcept]
```

Creates a forward\_list with no elements.

##### Parameters

|                                   |                      |
|-----------------------------------|----------------------|
| <a href="#"><code>__al</code></a> | An allocator object. |
|-----------------------------------|----------------------|

#### forward\_list() [3/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    const forward_list< _Tp, _Alloc > & __list,
    const __type_identity_t< _Alloc > & __al) [inline]
```

Copy constructor with allocator argument.

##### Parameters

|                                     |                      |
|-------------------------------------|----------------------|
| <a href="#"><code>__list</code></a> | Input list to copy.  |
| <a href="#"><code>__al</code></a>   | An allocator object. |

#### forward\_list() [4/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    forward_list< _Tp, _Alloc > && __list,
    const __type_identity_t< _Alloc > & __al) [inline], [noexcept]
```

Move constructor with allocator argument.

##### Parameters

|                                     |                      |
|-------------------------------------|----------------------|
| <a href="#"><code>__list</code></a> | Input list to move.  |
| <a href="#"><code>__al</code></a>   | An allocator object. |

#### forward\_list() [5/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
```

```

size_type __n,
const _Alloc & __al = _Alloc()) [inline], [explicit]

```

Creates a `forward_list` with default constructed elements.

#### Parameters

|                   |                                             |
|-------------------|---------------------------------------------|
| <code>__n</code>  | The number of elements to initially create. |
| <code>__al</code> | An allocator object.                        |

This constructor creates the `forward_list` with `__n` default constructed elements.

#### `forward_list()` [6/10]

```

template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    size_type __n,
    const _Tp & __value,
    const _Alloc & __al = _Alloc()) [inline]

```

Creates a `forward_list` with copies of an exemplar element.

#### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__al</code>    | An allocator object.                        |

This constructor fills the `forward_list` with `__n` copies of `__value`.

#### `forward_list()` [7/10]

```

template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::forward_list< _Tp, _Alloc >::forward_list (
    _InputIterator __first,
    _InputIterator __last,
    const _Alloc & __al = _Alloc()) [inline]

```

Builds a `forward_list` from a range.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__al</code>    | An allocator object. |

Create a `forward_list` consisting of copies of the elements from `[__first, __last)`. This is linear in `N` (where `N` is `distance(__first, __last)`).

#### `forward_list()` [8/10]

```

template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    const forward_list< _Tp, _Alloc > & __list) [inline]

```

The `forward_list` copy constructor.

## Parameters

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

**forward\_list()** [9/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    forward_list< _Tp, _Alloc > && ) [default]
```

The forward\_list move constructor.

## Parameters

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

The newly-created forward\_list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified forward\_list.

**forward\_list()** [10/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    std::initializer_list< _Tp > __il,
    const _Alloc & __al = _Alloc()) [inline]
```

Builds a forward\_list from an initializer\_list.

## Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__il</code> | An initializer_list of value_type. |
| <code>__al</code> | An allocator object.               |

Create a forward\_list consisting of copies of the elements in the initializer\_list `__il`. This is linear in `__il.size()`.

**~forward\_list()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::~forward_list () [inline], [noexcept]
```

The forward\_list dtor.

**5.434.3 Member Function Documentation****assign()** [1/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::forward_list< _Tp, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Assigns a range to a forward\_list.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a `forward_list` with copies of the elements in the range `[ __first, __last)`.

Note that the assignment completely changes the `forward_list` and that the number of elements of the resulting `forward_list` is the same as the number of elements assigned.

Referenced by `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign()`, `operator=()`, `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::operator=()` and `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::operator=()`.

**assign()** [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::assign (
    size_type __n,
    const _Tp & __val) [inline]
```

Assigns a given value to a `forward_list`.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a `forward_list` with `__n` copies of the given value. Note that the assignment completely changes the `forward_list`, and that the resulting `forward_list` has `__n` elements.

**assign()** [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::assign (
    std::initializer_list< _Tp > __il) [inline]
```

Assigns an `initializer_list` to a `forward_list`.

## Parameters

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <code>__il</code> | An <code>initializer_list</code> of <code>value_type</code> . |
|-------------------|---------------------------------------------------------------|

Replace the contents of the `forward_list` with copies of the elements in the `initializer_list` `__il`. This is linear in `__il.size()`.

**before\_begin()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::before_begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

**before\_begin()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::before_begin () [inline], [noexcept]
```

Returns a read/write iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::assign()`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::insert_after()`, `insert_after()`, `resize()`, and `resize()`.

### **begin()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

### **begin()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc>::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::assign()`, and `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::push_back()`.

### **cbegin\_begin()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::cbegin_begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::assign()`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::emplace_front()`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::push_back()`, and `remove()`.

### **cbegin()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by `std::operator==()`.

### **cend()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by `std::operator==()`.

### **clear()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::clear () [inline], [noexcept]
```

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::assign()`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::operator=()`, and `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::operator=()`.

**emplace\_after()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
iterator std::forward_list<_Tp, _Alloc >::emplace_after (
    const_iterator __pos,
    _Args &&... __args) [inline]
```

Constructs object in forward\_list after the specified iterator.

**Parameters**

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos</code>  | A const_iterator into the forward_list. |
| <code>__args</code> | Arguments.                              |

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type T constructed with `T(std::forward<Args>(args) ...)` after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

**emplace\_front()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
reference std::forward_list<_Tp, _Alloc >::emplace_front (
    _Args &&... __args) [inline]
```

Constructs object in forward\_list at the front of the list.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__args</code> | Arguments. |
|---------------------|------------|

This function will insert an object of type Tp constructed with `Tp(std::forward<Args>(args) ...)` at the front of the list Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

**empty()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
bool std::forward_list<_Tp, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the forward\_list is empty. (Thus begin() would equal end().)

Referenced by [insert\\_after\(\)](#), and [sort\(\)](#).

**end()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.



**end()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign()`, `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::insert_after()`, `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::insert_after()`, `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::resize()`, and `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::resize()`.

**erase\_after()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
    const_iterator __pos) [inline]
```

Removes the element pointed to by the iterator following `pos`.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__pos</code> | Iterator pointing before element to be erased. |
|--------------------|------------------------------------------------|

**Returns**

An iterator pointing to the element following the one that was erased, or `end()` if no such element exists.

This function will erase the element at the given position and thus shorten the `forward_list` by one.

Due to the nature of a `forward_list` this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign()`, `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::insert_after()`, `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::insert_after()`, `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::resize()`, and `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::resize()`.

**erase\_after()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
    const_iterator __pos,
    const_iterator __last) [inline]
```

Remove a range of elements.

**Parameters**

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>__pos</code>  | Iterator pointing before the first element to be erased.     |
| <code>__last</code> | Iterator pointing to one past the last element to be erased. |

**Returns**

`__last`

This function will erase the elements in the range `(__pos, __last)` and shorten the `forward_list` accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed.

The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**front()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
reference std::forward\_list<\_Tp, \_Alloc>::front\(\) [inline]
```

Returns a read/write reference to the data at the first element of the `forward_list`.  
Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::emplace\\_front\(\)](#).

**front()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_reference std::forward\_list<\_Tp, \_Alloc>::front\(\) const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the `forward_list`.

**get\_allocator()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
allocator_type std::forward\_list<\_Tp, \_Alloc>::get\_allocator\(\) const [inline], [noexcept]
```

Get a copy of the memory allocation object.  
Referenced by [insert\\_after\(\)](#), [insert\\_after\(\)](#), and [remove\(\)](#).

**insert\_after()** [1/4]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator, typename>
forward\_list<\_Tp, \_Alloc>::iterator forward\_list::insert\_after (
    const_iterator __pos,
    _InputIterator __first,
    _InputIterator __last)
```

Inserts a range into the `forward_list`.

**Parameters**

|                      |                                                  |
|----------------------|--------------------------------------------------|
| <code>__pos</code>   | An iterator into the <code>forward_list</code> . |
| <code>__first</code> | An input iterator.                               |
| <code>__last</code>  | An input iterator.                               |

**Returns**

An iterator pointing to the last inserted element or `__pos` if `__first == __last`.

This function will insert copies of the data in the range `[ __first, __last)` into the `forward_list` after the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

References [forward\\_list\(\)](#), [before\\_begin\(\)](#), [empty\(\)](#), [end\(\)](#), and [get\\_allocator\(\)](#).

**insert\_after()** [2/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward\_list<\_Tp, \_Alloc>::insert\_after (
    const_iterator __pos,
    const _Tp & __val) [inline]
```

Inserts given value into `forward_list` after specified iterator.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | An iterator into the <code>forward_list</code> . |
| <code>__val</code> | Data to be inserted.                             |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::assign\(\)](#), [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\\_after\(\)](#), and [resize\(\)](#).

**insert\_after()** [3/4]

```
template<typename _Tp, typename _Alloc>
forward_list<_Tp, _Alloc>::iterator forward_list::insert_after (
    const_iterator __pos,
    size_type __n,
    const _Tp & __val)
```

Inserts a number of copies of given data into the `forward_list`.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | An iterator into the <code>forward_list</code> . |
| <code>__n</code>   | Number of elements to be inserted.               |
| <code>__val</code> | Data to be inserted.                             |

**Returns**

An iterator pointing to the last inserted copy of `val` or `pos` if `n == 0`.

This function will insert a specified number of copies of the given data after the location specified by `pos`. This operation is linear in the number of elements inserted and does not invalidate iterators and references. References [forward\\_list\(\)](#), [before\\_begin\(\)](#), [end\(\)](#), and [get\\_allocator\(\)](#).

**insert\_after()** [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc>::insert_after (
    const_iterator __pos,
    std::initializer_list<_Tp> __il) [inline]
```

Inserts the contents of an `initializer_list` into `forward_list` after the specified iterator.

**Parameters**

|                    |                                                               |
|--------------------|---------------------------------------------------------------|
| <code>__pos</code> | An iterator into the <code>forward_list</code> .              |
| <code>__il</code>  | An <code>initializer_list</code> of <code>value_type</code> . |

**Returns**

An iterator pointing to the last inserted element or `__pos` if `__il` is empty.

This function will insert copies of the data in the `initializer_list` `__il` into the `forward_list` before the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

**max\_size()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
size_type std::forward\_list<\_Tp, \_Alloc >::max\_size () const [inline], [noexcept]
```

Returns the largest possible number of elements of `forward_list`.

**merge()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward\_list<\_Tp, \_Alloc >::merge (
    forward\_list<\_Tp, \_Alloc > && \_\_list) [inline]
```

Merge sorted lists.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__list</code> | Sorted list to merge. |
|---------------------|-----------------------|

Assumes that both `__list` and this list are sorted according to `operator<()`. Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equal.

Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp > >::merge\(\)](#).

**merge()** [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _Comp>
void forward\_list::merge (
    forward\_list<\_Tp, \_Alloc > && \_\_list,
    \_Comp \_\_comp)
```

Merge sorted lists according to comparison function.

**Parameters**

|                     |                                          |
|---------------------|------------------------------------------|
| <code>__list</code> | Sorted list to merge.                    |
| <code>__comp</code> | Comparison function defining sort order. |

Assumes that both `__list` and this list are sorted according to `comp`. Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equivalent according to `comp()`.

References [forward\\_list\(\)](#), [std::\\_\\_addressof\(\)](#), and [std::move\(\)](#).

**operator=()** [1/3]

```
template<typename _Tp, typename _Alloc>
forward\_list<\_Tp, \_Alloc > & forward\_list::operator= (
    const forward\_list<\_Tp, \_Alloc > & \_\_list)
```

The `forward_list` assignment operator.

**Parameters**

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__list</code> | A <code>forward_list</code> of identical element and allocator types. |
|---------------------|-----------------------------------------------------------------------|

All the elements of `__list` are copied.

Whether the allocator is copied depends on the allocator traits.

References [forward\\_list\(\)](#), [std::\\_\\_addressof\(\)](#), [assign\(\)](#), and [clear\(\)](#).

**operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list & std::forward_list< _Tp, _Alloc >::operator= (
    forward_list< _Tp, _Alloc > && __list) [inline], [noexcept]
```

The forward\_list move assignment operator.

**Parameters**

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

The contents of `__list` are moved into this forward\_list (without copying, if the allocators permit it).

Afterwards `__list` is a valid, but unspecified forward\_list

Whether the allocator is moved depends on the allocator traits.

**operator=()** [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list & std::forward_list< _Tp, _Alloc >::operator= (
    std::initializer_list< _Tp > __il) [inline]
```

The forward\_list initializer list assignment operator.

**Parameters**

|                   |                                    |
|-------------------|------------------------------------|
| <code>__il</code> | An initializer_list of value_type. |
|-------------------|------------------------------------|

Replace the contents of the forward\_list with copies of the elements in the initializer\_list `__il`. This is linear in `__il.size()`.

**pop\_front()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::pop_front () [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the forward\_list by one. Due to the nature of a forward\_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

**push\_front()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::push_front (
    const _Tp & __val) [inline]
```

Add data to the front of the forward\_list.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__val</code> | Data to be added. |
|--------------------|-------------------|

This is a typical stack operation. The function creates an element at the front of the forward\_list and assigns the given data to it. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

**remove()**

```
template<typename _Tp, typename _Alloc>
auto forward_list::remove (
    const _Tp & __val)
```

Remove all elements equal to value.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__val</code> | The value to remove. |
|--------------------|----------------------|

Removes every element in the list equal to `__val`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [forward\\_list\(\)](#), [cbefore\\_begin\(\)](#), [get\\_allocator\(\)](#), and [splice\\_after\(\)](#).

**remove\_if()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _Pred>
__remove_return_type std::forward_list<_Tp, _Alloc >::remove_if (
    _Pred __pred)
```

Remove all elements satisfying a predicate.

**Parameters**

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__pred</code> | Unary predicate function or object. |
|---------------------|-------------------------------------|

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**resize()** [1/2]

```
template<typename _Tp, typename _Alloc>
void forward_list::resize (
    size_type __sz)
```

Resizes the `forward_list` to the specified number of elements.

**Parameters**

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__sz</code> | Number of elements the <code>forward_list</code> should contain. |
|-------------------|------------------------------------------------------------------|

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and the new elements are default constructed.

References [forward\\_list\(\)](#), [before\\_begin\(\)](#), [end\(\)](#), and [erase\\_after\(\)](#).

**resize()** [2/2]

```
template<typename _Tp, typename _Alloc>
void forward_list::resize (
    size_type __sz,
    const value_type & __val)
```

Resizes the `forward_list` to the specified number of elements.

## Parameters

|                    |                                                                  |
|--------------------|------------------------------------------------------------------|
| <code>__sz</code>  | Number of elements the <code>forward_list</code> should contain. |
| <code>__val</code> | Data with which new elements should be populated.                |

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and new elements are populated with given data.

References [forward\\_list\(\)](#), [before\\_begin\(\)](#), [end\(\)](#), [erase\\_after\(\)](#), and [insert\\_after\(\)](#).

**reverse()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::reverse () [inline], [noexcept]
```

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

**sort()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::sort () [inline]
```

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Referenced by [std::forward\\_list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::sort\(\)](#).

**sort()** [2/2]

```
template<typename _Tp, class _Alloc>
template<typename _Comp>
void forward_list::sort (
    _Comp __comp)
```

Sort the `forward_list` using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

References [forward\\_list\(\)](#), and [empty\(\)](#).

**splice\_after()** [1/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
    const_iterator __pos,
    forward_list< _Tp, _Alloc > && ,
    const_iterator __before,
    const_iterator __last) [inline], [noexcept]
```

Insert range from another `forward_list`.

## Parameters

|                       |                                                                         |
|-----------------------|-------------------------------------------------------------------------|
| <code>__pos</code>    | Iterator referencing the element to insert after.                       |
| <code>__list</code>   | Source list.                                                            |
| <code>__before</code> | Iterator referencing before the start of range in <code>__list</code> . |
| <code>__last</code>   | Iterator referencing the end of range in <code>__list</code> .          |

Removes elements in the range `(__before, __last)` and inserts them after `__pos` in constant time.

Undefined if `__pos` is in `(__before, __last)`.

**splice\_after()** [2/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc >::splice_after (
    const_iterator __pos,
    forward_list<_Tp, _Alloc > && __list) [inline], [noexcept]
```

Insert contents of another forward\_list.

**Parameters**

|                     |                                                   |
|---------------------|---------------------------------------------------|
| <code>__pos</code>  | Iterator referencing the element to insert after. |
| <code>__list</code> | Source list.                                      |

The elements of `list` are inserted in constant time after the element referenced by `pos`. `list` becomes an empty list.

Requires `this != &x`.

Referenced by [remove\(\)](#), and [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::splice\\_after\(\)](#).

**splice\_after()** [3/4]

```
template<typename _Tp, typename _Alloc>
void forward_list::splice_after (
    const_iterator __pos,
    forward_list<_Tp, _Alloc > && __list,
    const_iterator __i) [noexcept]
```

Insert element from another forward\_list.

**Parameters**

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>__pos</code>  | Iterator referencing the element to insert after.            |
| <code>__list</code> | Source list.                                                 |
| <code>__i</code>    | Iterator referencing the element before the element to move. |

Removes the element in `list` referenced by `__i` and inserts it into the current list after `__pos`.

References [forward\\_list\(\)](#).

**splice\_after()** [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc >::splice_after (
    const_iterator __pos,
    forward_list<_Tp, _Alloc > & ,
    const_iterator __before,
    const_iterator __last) [inline], [noexcept]
```

Insert range from another forward\_list.

**Parameters**

|                       |                                                                         |
|-----------------------|-------------------------------------------------------------------------|
| <code>__pos</code>    | Iterator referencing the element to insert after.                       |
| <code>__list</code>   | Source list.                                                            |
| <code>__before</code> | Iterator referencing before the start of range in <code>__list</code> . |
| <code>__last</code>   | Iterator referencing the end of range in <code>__list</code> .          |

Removes elements in the range `(__before, __last)` and inserts them after `__pos` in constant time.

Undefined if `__pos` is in `(__before, __last)`.



**swap()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::swap (
    forward_list< _Tp, _Alloc > & __list) [inline], [noexcept]
```

Swaps data with another forward\_list.

**Parameters**

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <code>__list</code> | A forward_list of the same element and allocator types. |
|---------------------|---------------------------------------------------------|

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is overloaded such that `std::swap(l1, l2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

**unique()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
__remove_return_type std::forward_list< _Tp, _Alloc >::unique () [inline]
```

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::unique()`.

**unique()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _BinPred>
__remove_return_type std::forward_list< _Tp, _Alloc >::unique (
    _BinPred __binary_pred)
```

Remove consecutive elements satisfying a predicate.

**Parameters**

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>__binary_pred</code> | Binary predicate function or object. |
|----------------------------|--------------------------------------|

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

The documentation for this class was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

**5.435 std::fpos< \_StateT > Class Template Reference**

```
#include <postypes.h>
```

**Public Member Functions**

- `fpos` (const `fpos` &)=default
- `fpos` (streamoff \_\_off)
- `operator streamoff` () const

- `fpos operator+ (streamoff __off) const`
- `fpos & operator+= (streamoff __off)`
- `streamoff operator- (const fpos & __other) const`
- `fpos operator- (streamoff __off) const`
- `fpos & operator-= (streamoff __off)`
- `fpos & operator= (const fpos &)=default`
- `_StateT state () const`
- `void state (_StateT __st)`

### 5.435.1 Detailed Description

```
template<typename _StateT>
class std::fpos<_StateT>
```

Class representing stream positions.

The standard places no requirements upon the template parameter StateT. In this implementation StateT must be DefaultConstructible, CopyConstructible and Assignable. The standard only requires that fpos should contain a member of type StateT. In this implementation it also contains an offset stored as a signed integer.

#### Parameters

|               |                                           |
|---------------|-------------------------------------------|
| <i>StateT</i> | Type passed to and returned from state(). |
|---------------|-------------------------------------------|

### 5.435.2 Constructor & Destructor Documentation

#### fpos()

```
template<typename _StateT>
std::fpos<_StateT>::fpos (
    streamoff __off) [inline]
```

Construct position from offset.

### 5.435.3 Member Function Documentation

#### operator streamoff()

```
template<typename _StateT>
std::fpos<_StateT>::operator streamoff () const [inline]
```

Convert to streamoff.

#### operator+()

```
template<typename _StateT>
fpos std::fpos<_StateT>::operator+ (
    streamoff __off) const [inline]
```

Add position and offset.

#### operator+=()

```
template<typename _StateT>
fpos & std::fpos<_StateT>::operator+= (
    streamoff __off) [inline]
```

Add offset to this position.

**operator-() [1/2]**

```
template<typename _StateT>
streamoff std::fpos< _StateT >::operator- (
    const fpos< _StateT > & __other) const [inline]
```

Subtract position to return offset.

**operator-() [2/2]**

```
template<typename _StateT>
fpos std::fpos< _StateT >::operator- (
    streamoff __off) const [inline]
```

Subtract offset from position.

**operator-=( )**

```
template<typename _StateT>
fpos & std::fpos< _StateT >::operator-= (
    streamoff __off) [inline]
```

Subtract offset from this position.

**state() [1/2]**

```
template<typename _StateT>
_StateT std::fpos< _StateT >::state () const [inline]
```

Return the last set value of *st*.

**state() [2/2]**

```
template<typename _StateT>
void std::fpos< _StateT >::state (
    _StateT __st) [inline]
```

Remember the value of *st*.

The documentation for this class was generated from the following file:

- [postypes.h](#)

**5.436 \_\_gnu\_cxx::free\_list Class Reference**

```
#include <bitmap_allocator.h>
```

Inheritance diagram for `__gnu_cxx::free_list`:



## Public Types

- typedef `__mutex` `__mutex_type`
- typedef `vector_type::iterator` `iterator`
- typedef `std::size_t * value_type`
- typedef `__detail::__mini_vector` `< value_type > vector_type`

## Public Member Functions

- void `_M_clear` ()
- `std::size_t * _M_get` (`std::size_t __sz`)
- void `_M_insert` (`std::size_t * __addr`) throw ()

### 5.436.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the `bitmap_allocator`.

### 5.436.2 Member Function Documentation

#### `_M_clear()`

```
void __gnu_cxx::free_list::_M_clear ()
```

This function just clears the internal Free List, and gives back all the memory to the OS.

#### `_M_get()`

```
std::size_t * __gnu_cxx::free_list::_M_get (
    std::size_t __sz)
```

This function gets a block of memory of the specified size from the free list.

#### Parameters

|                   |                                           |
|-------------------|-------------------------------------------|
| <code>__sz</code> | The size in bytes of the memory required. |
|-------------------|-------------------------------------------|

#### Returns

A pointer to the new memory block of size at least equal to that requested.

#### `_M_insert()`

```
void __gnu_cxx::free_list::_M_insert (
    std::size_t * __addr) throw ( ) [inline]
```

This function returns the block of memory to the internal free list.

#### Parameters

|                     |                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------|
| <code>__addr</code> | The pointer to the memory block that was given by a call to the <code>_M_get</code> function. |
|---------------------|-----------------------------------------------------------------------------------------------|

Referenced by `__gnu_cxx::bitmap_allocator< _Tp1 >::_M_deallocate_single_object()`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

### 5.437 std::from\_chars\_result Struct Reference

```
#include <charconv>
```

#### Public Attributes

- `errc ec`
- `const char * ptr`

#### Friends

- `bool operator== (const from_chars_result &, const from_chars_result &)=default`

#### 5.437.1 Detailed Description

Result type of `std::from_chars`.

The documentation for this struct was generated from the following file:

- [charconv](#)

### 5.438 std::front\_insert\_iterator< \_Container > Class Template Reference

```
#include <stl_iterator.h>
```

Inheritance diagram for `std::front_insert_iterator< _Container >`:



#### Public Types

- `typedef _Container container_type`
- `using difference_type`
- `typedef output_iterator_tag iterator_category`
- `typedef void pointer`
- `typedef void reference`
- `typedef void value_type`

**Public Member Functions**

- constexpr `front_insert_iterator` (`_Container &__x`)
- constexpr `front_insert_iterator & operator*` ()
- constexpr `front_insert_iterator & operator++` ()
- constexpr `front_insert_iterator operator++` (int)
- constexpr `front_insert_iterator & operator=` (const typename `_Container::value_type &__value`)
- constexpr `front_insert_iterator & operator=` (typename `_Container::value_type &&__value`)

**Protected Attributes**

- `_Container * container`

**5.438.1 Detailed Description**

```
template<typename _Container>
class std::front_insert_iterator<_Container>
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using `push_front`.

Tip: Using the `front_inserter` function to create these iterators can save typing.

**5.438.2 Member Typedef Documentation****container\_type**

```
template<typename _Container>
typedef _Container std::front_insert_iterator<_Container>::container_type
```

A nested typedef for the type of whatever container you used.

**iterator\_category**

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category [inherited]
```

One of the [tag types](#).

**pointer**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

**reference**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value\_type.

**value\_type**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
```

The type "pointed to" by the iterator.

### 5.438.3 Constructor & Destructor Documentation

#### front\_insert\_iterator()

```
template<typename _Container>
std::front_insert_iterator< _Container >::front_insert_iterator (
    _Container & __x) [inline], [explicit], [constexpr]
```

The only way to create this iterator is with a container.

References [std::\\_\\_addressof\(\)](#).

### 5.438.4 Member Function Documentation

#### operator\*()

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator* () [inline], [nodiscard],
[constexpr]
```

Simply returns *\*this*.

#### operator++() [1/2]

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator++ () [inline], [constexpr]
```

Simply returns *\*this*. (This iterator does not *move*.)

#### operator++() [2/2]

```
template<typename _Container>
front_insert_iterator std::front_insert_iterator< _Container >::operator++ (
    int ) [inline], [constexpr]
```

Simply returns *\*this*. (This iterator does not *move*.)

#### operator=()

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator= (
    const typename _Container::value_type & __value) [inline], [constexpr]
```

#### Parameters

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

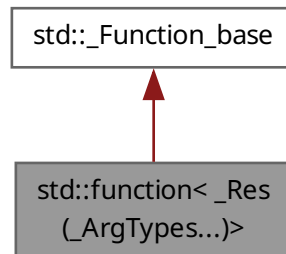
The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

**5.439 std::function< \_Res(\_ArgTypes...)> Class Template Reference**

```
#include <std_function.h>
```

Inheritance diagram for std::function< \_Res(\_ArgTypes...)>:

**Public Types**

- typedef `_Res` **result\_type**

**Public Member Functions**

- `function` () noexcept
- template<typename \_Functor, typename \_Constraints = \_Requires< \_Callable<\_Functor>>>>  
`function` (\_Functor &&\_\_f) noexcept(\_Handler<\_Functor>::template \_S\_nothrow\_init<\_Functor>())
- `function` (const function &\_\_x)
- `function` (function &&\_\_x) noexcept
- `function` (nullptr\_t) noexcept
- `operator bool` () const noexcept
- `_Res operator()` (\_ArgTypes... \_\_args) const
- template<typename \_Functor>  
`_Requires< _Callable<_Functor>, function &> operator=` (\_Functor &&\_\_f) noexcept(\_Handler<\_Functor>::template \_S\_nothrow\_init<\_Functor>())
- `function & operator=` (const `function` &\_\_x)
- `function & operator=` (function &&\_\_x) noexcept
- `function & operator=` (nullptr\_t) noexcept
- template<typename \_Functor>  
`function & operator=` (reference\_wrapper<\_Functor> \_\_f) noexcept
- void `swap` (function &\_\_x) noexcept
- const `type_info` & `target_type` () const noexcept
- template<typename \_Functor>  
`_Functor * target` () noexcept
- template<typename \_Functor>  
const `_Functor * target` () const noexcept



### 5.439.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes>
class std::function< _Res(_ArgTypes...)>
```

Polymorphic function wrapper.

Since

C++11

### 5.439.2 Constructor & Destructor Documentation

#### function() [1/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function () [inline], [noexcept]
Default construct creates an empty function call wrapper.
```

#### Postcondition

```
!(bool)*this
```

References [function\(\)](#).

Referenced by [function\(\)](#), [function\(\)](#), [function\(\)](#), [function\(\)](#), [function\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [swap\(\)](#), and [target\(\)](#).

#### function() [2/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
    nullptr_t ) [inline], [noexcept]
```

Creates an empty function call wrapper.

#### Postcondition

```
!(bool)*this
```

References [function\(\)](#).

#### function() [3/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
    const function< _Res(_ArgTypes...)> & __x) [inline]
```

Function copy constructor.

#### Parameters

|                                                                                                     |                                                  |
|-----------------------------------------------------------------------------------------------------|--------------------------------------------------|
| <a href="#"></a> | A function object with identical call signature. |
| <a href="#"></a> |                                                  |

#### Postcondition

```
bool(*this) == bool(__x)
```

The newly-created function contains a copy of the target of `__x` (if it has one).

References [function\(\)](#).

**function()** [4/5]

```
template<typename _Res, typename... _ArgTypes>
std::function<_Res(_ArgTypes...)>::function (
    function<_Res(_ArgTypes...)> && __x) [inline], [noexcept]
```

Function move constructor.

**Parameters**

|                                                                                                    |                                                         |
|----------------------------------------------------------------------------------------------------|---------------------------------------------------------|
|                   | A function object rvalue with identical call signature. |
|  <code>__x</code> |                                                         |

The newly-created function contains the target of `__x` (if it has one).

References [function\(\)](#).

**function()** [5/5]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor, typename _Constraints = _Requires<_Callable<_Functor>>>
std::function<_Res(_ArgTypes...)>::function (
    _Functor && __f) [inline], [noexcept]
```

Builds a function that targets a copy of the incoming function object.

**Parameters**

|                                                                                                    |                                                                                                                                              |
|----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
|                   | A function object that is callable with parameters of type <code>ArgTypes . . .</code> and returns a value convertible to <code>Res</code> . |
|                  |                                                                                                                                              |
|                 |                                                                                                                                              |
|                 |                                                                                                                                              |
|  <code>f</code> |                                                                                                                                              |

The newly-created function object will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a null function pointer, null pointer-to-member, or empty `std::function`, the newly-created object will be empty.

If `__f` is a non-null function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

References [function\(\)](#), and [std::forward\(\)](#).

**5.439.3 Member Function Documentation****operator bool()**

```
template<typename _Res, typename... _ArgTypes>
std::function<_Res(_ArgTypes...)>::operator bool () const [inline], [explicit], [noexcept]
```

Determine if the function wrapper has a target.

**Returns**

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw exceptions.

**operator()()**

```
template<typename _Res, typename... _ArgTypes>
_Res std::function<_Res(_ArgTypes...)>::operator() (
    _ArgTypes... __args) const [inline]
```

Invokes the function targeted by `*this`.

**Returns**

the result of the target.

**Exceptions**

|                                  |                                |
|----------------------------------|--------------------------------|
| <code>`bad_function_call`</code> | when <code>!(bool)*this</code> |
|----------------------------------|--------------------------------|

The function call operator invokes the target function object stored by `this`.

References [std::forward\(\)](#), and [operator\(\)](#).

Referenced by [operator\(\)](#).

**operator=()** [1/5]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
_requires<_Callable<_Functor>, function &> std::function<_Res(_ArgTypes...)>::operator= (
    _Functor && __f) [inline], [noexcept]
```

Function assignment to a new target.

**Parameters**

|                |                                                                                                                         |
|----------------|-------------------------------------------------------------------------------------------------------------------------|
| ↩              | A function object that is callable with parameters of type <code>_ArgTypes...</code> and returns a value convertible to |
| ↩              | <code>_Res</code> .                                                                                                     |
| ↩              |                                                                                                                         |
| ↩              |                                                                                                                         |
| <code>f</code> |                                                                                                                         |

**Returns**

`*this`

**Since**

C++11

This function object wrapper will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a null function pointer or null pointer-to-member, this object will be empty.

If `__f` is a non-null function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

References [function\(\)](#), [std::forward\(\)](#), and [operator=\(\)](#).

**operator=()** [2/5]

```
template<typename _Res, typename... _ArgTypes>
function & std::function<_Res(_ArgTypes...)>::operator= (
    const function<_Res(_ArgTypes...)> & __x) [inline]
```

Function assignment operator.

**Parameters**

|                |                                           |
|----------------|-------------------------------------------|
| ↩              | A function with identical call signature. |
| <code>x</code> |                                           |

**Postcondition**

```
(bool)*this == (bool)x
```

**Returns**

```
*this
```

The target of `__x` is copied to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

References [function\(\)](#), and [operator=\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

**operator=()** [3/5]

```
template<typename _Res, typename... _ArgTypes>
function & std::function<_Res(_ArgTypes...)>::operator= (
    function<_Res(_ArgTypes...)> && __x) [inline], [noexcept]
```

Function move-assignment operator.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A function rvalue with identical call signature. |
|------------------|--------------------------------------------------|

**Returns**

```
*this
```

The target of `__x` is moved to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

References [function\(\)](#), [std::move\(\)](#), and [operator=\(\)](#).

**operator=()** [4/5]

```
template<typename _Res, typename... _ArgTypes>
function & std::function<_Res(_ArgTypes...)>::operator= (
    nullptr_t ) [inline], [noexcept]
```

Function assignment to empty.

**Postcondition**

```
!(bool)*this
```

**Returns**

```
*this
```

The target of `*this` is deallocated, leaving it empty.

References [function\(\)](#), and [operator=\(\)](#).

**operator=()** [5/5]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
function & std::function<_Res(_ArgTypes...)>::operator= (
    reference_wrapper<_Functor> & __f) [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

References [function\(\)](#), and [operator=\(\)](#).

**swap()**

```
template<typename _Res, typename... _ArgTypes>
void std::function< _Res(_ArgTypes...)>::swap (
    function< _Res(_ArgTypes...)> & __x) [inline], [noexcept]
```

Swap the targets of two function objects.

**Parameters**

|                  |                                           |
|------------------|-------------------------------------------|
| <code>__x</code> | A function with identical call signature. |
|------------------|-------------------------------------------|

Swap the targets of `this` function object and `__f`. This function will not throw exceptions.

References [function\(\)](#), and [swap\(\)](#).

Referenced by [swap\(\)](#).

**target() [1/2]**

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
const _Functor * std::function< _Res(_ArgTypes...)>::target () const [inline], [noexcept]
```

Access the stored target function object.

**Returns**

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a null pointer.

This function does not throw exceptions.

References [target\(\)](#), and [target\\_type\(\)](#).

**target() [2/2]**

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
_Functor * std::function< _Res(_ArgTypes...)>::target () [inline], [noexcept]
```

Access the stored target function object.

**Returns**

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a null pointer.

This function does not throw exceptions.

References [function\(\)](#), and [target\(\)](#).

Referenced by [target\(\)](#), and [target\(\)](#).

**target\_type()**

```
template<typename _Res, typename... _ArgTypes>
const type\_info & std::function< _Res(_ArgTypes...)>::target_type () const [inline], [noexcept]
```

Determine the type of the target of this function object wrapper.

## Returns

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw exceptions.

References [target\\_type\(\)](#).

Referenced by [target\(\)](#), and [target\\_type\(\)](#).

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

5.440 `std::function_ref<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>` Class Template Reference

```
#include <functional>
```

## Public Member Functions

- `template<typename _Fn, typename _Vt = remove_reference_t<_Fn>>`  
requires `(lis_same_v<remove_cv_t<_Vt>, function_ref>) && (lis_member_pointer_v<_Vt>) && (lis_function_v<_Vt>) && __is_invocable_using<_Vt _GLIBCXX_MOF_CV>`  
`constexpr function_ref (_Fn &&__f) noexcept`
- `template<typename _Fn>`  
requires `is_function_v<_Fn> && __is_invocable_using<_Fn*>`  
`function_ref (_Fn *__fn) noexcept`
- `template<auto __fn>`  
requires `__is_invocable_using<const decltype(__fn)>`  
`constexpr function_ref (nontype_t< __fn >) noexcept`
- `template<auto __fn, typename _Td>`  
requires `__is_invocable_using<const decltype(__fn)&, _Td _GLIBCXX_MOF_CV*>`  
`constexpr function_ref (nontype_t< __fn >, _Td _GLIBCXX_MOF_CV *__ptr) noexcept`
- `template<auto __fn, typename _Up, typename _Td = remove_reference_t<_Up>>`  
requires `(lis_rvalue_reference_v<_Up&&>) && __is_invocable_using<const decltype(__fn)&, _Td _GLIBCXX_MOF_CV>`  
`constexpr function_ref (nontype_t< __fn >, _Up &&__ref) noexcept`
- `_Res operator() (_ArgTypes... __args) const noexcept(_Noex)`
- `template<typename _Tp>`  
requires `(lis_same_v<_Tp, function_ref>) && (lis_pointer_v<_Tp>) && (!__is_nontype_v<_Tp>)`  
`function_ref & operator= (_Tp)=delete`

## 5.440.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
class std::function_ref<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
```

Non-owning polymorphic function wrapper.

Since

C++26

The `std::function_ref` class template is a non-owning call wrapper, that refers to a bound object. Using `function_ref` outside of the lifetime of the bound object has undefined behavior.

It supports const-qualification and no-throw guarantees. The qualifications and exception-specification of the signature are respected when invoking the reference function.

## 5.440.2 Constructor & Destructor Documentation

### function\_ref() [1/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn>
requires is_function_v<_Fn> && __is_invocable_using<_Fn*>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
    _Fn * __fn) [inline], [noexcept]
```

Target and bound object is function pointed by parameter.

### function\_ref() [2/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = remove_reference_t<_Fn>>
requires (!is_same_v<remove_cv_t<_Vt>, function_ref>) && (!is_member_pointer_v<_Vt>) && (!is_
function_v<_Vt>) && __is_invocable_using<_Vt _GLIBCXX_MOF_CV&>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
    _Fn && __f) [inline], [constexpr], [noexcept]
```

Target and bound object is object referenced by parameter.

References [std::addressof\(\)](#).

### function\_ref() [3/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn>
requires __is_invocable_using<const decltype(__fn)&>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
    nontype_t< __fn > ) [inline], [constexpr], [noexcept]
```

Target object is \_\_fn. There is no bound object.

### function\_ref() [4/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Up, typename _Td = remove_reference_t<_Up>>
requires (!is_rvalue_reference_v<_Up&&>) && __is_invocable_using<const decltype(__fn)&, _Td &
GLIBCXX_MOF_CV&>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
    nontype_t< __fn > ,
    _Up && __ref) [inline], [constexpr], [noexcept]
```

Target object is equivalent to `std::bind_front<_fn>(std::ref(__ref))`. Bound object is object referenced by second parameter.

References [std::addressof\(\)](#).

### function\_ref() [5/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Td>
requires __is_invocable_using<const decltype(__fn)&, _Td _GLIBCXX_MOF_CV*>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
    nontype_t< __fn > ,
    _Td _GLIBCXX_MOF_CV * __ptr) [inline], [constexpr], [noexcept]
```

Target object is equivalent to `std::bind_front<_fn>(__ptr)`. Bound object is object pointed by second parameter (if any).

### 5.440.3 Member Function Documentation

#### operator>()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
_Res std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
    _ArgTypes... __args) const [inline], [noexcept]
```

Invoke the target object.

The bound object will be invoked using the supplied arguments, and as const or non-const, as dictated by the template arguments of the `function_ref` specialization.

References [std::forward\(\)](#).

The documentation for this class was generated from the following file:

- [funcref\\_impl.h](#)

## 5.441 std::future<\_Res> Class Template Reference

```
#include <future>
```

Inheritance diagram for `std::future<_Res>`:



### Public Member Functions

- **future** (const [future](#) &)=delete
- **future** ([future](#) &&\_\_uf) noexcept
- [\\_Res](#) **get** ()
- **future** & **operator=** (const [future](#) &)=delete
- **future** & **operator=** ([future](#) &&\_\_fut) noexcept
- [shared\\_future](#)<[\\_Res](#)> **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- template<typename [\\_Rep](#), typename [\\_Period](#)>  
[future\\_status](#) **wait\_for** (const [chrono::duration](#)<[\\_Rep](#), [\\_Period](#)> &\_\_rel) const
- template<typename [\\_Clock](#), typename [\\_Duration](#)>  
[future\\_status](#) **wait\_until** (const [chrono::time\\_point](#)<[\\_Clock](#), [\\_Duration](#)> &\_\_abs) const

### Protected Types

- typedef `__future_base::_Result<_Res> & __result_type`



### Protected Member Functions

- `__result_type _M_get_result () const`
- `void _M_swap (__basic_future &__that) noexcept`

### Friends

- `template<typename _Fn, typename... _Args>  
future< __async_result_of< _Fn, _Args... > > async (launch, _Fn &&, _Args &&...)`
- `template<typename>  
class packaged_task`
- `class promise< _Res >`

#### 5.441.1 Detailed Description

```
template<typename _Res>  
class std::future< _Res >
```

Primary template for future.

#### 5.441.2 Constructor & Destructor Documentation

##### future()

```
template<typename _Res>  
std::future< _Res >::future (   
    future< _Res > && __uf) [inline], [noexcept]
```

Move constructor.

#### 5.441.3 Member Function Documentation

##### \_M\_get\_result()

```
template<typename _Res>  
__result_type std::__basic_future< _Res >::_M_get_result () const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

##### get()

```
template<typename _Res>  
_Res std::future< _Res >::get () [inline]
```

Retrieving the value.

The documentation for this class was generated from the following file:

- `future`

#### 5.442 std::future< \_Res & > Class Template Reference

```
#include <future>
```

Inheritance diagram for std::future<\_Res &>:



### Public Member Functions

- **future** (const [future](#) &)=delete
- **future** (const [future](#) &)=delete
- **future** ([future](#) &&\_\_uf) noexcept
- **future** ([future](#) &&\_\_uf) noexcept
- [\\_Res](#) **get** ()
- [\\_Res](#) & **get** ()
- **future** & **operator=** (const [future](#) &)=delete
- **future** & **operator=** (const [future](#) &)=delete
- **future** & **operator=** ([future](#) &&\_\_fut) noexcept
- **future** & **operator=** ([future](#) &&\_\_fut) noexcept
- [shared\\_future](#)<\_Res > **share** () noexcept
- [shared\\_future](#)<\_Res &> **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- [future\\_status](#) **wait\_for** (const [chrono::duration](#)<\_Rep, \_Period > &\_\_rel) const
- [future\\_status](#) **wait\_until** (const [chrono::time\\_point](#)<\_Clock, \_Duration > &\_\_abs) const

### Protected Types

- typedef \_\_future\_base::\_Result<\_Res > & \_\_**result\_type**

### Protected Member Functions

- \_\_result\_type **\_M\_get\_result** () const
- void **\_M\_swap** (\_\_[basic\\_future](#) &\_\_that) noexcept

### Friends

- [future](#)<\_\_async\_result\_of<\_Fn, \_Args... >> **async** ([launch](#), \_Fn &&, \_Args &&...)
- template<typename \_Fn, typename... \_Args>  
[future](#)<\_\_async\_result\_of<\_Fn, \_Args... >> **async** ([launch](#), \_Fn &&, \_Args &&...)
- class **packaged\_task**

- `template<typename>`  
  class **packaged\_task**
- class **promise**<\_Res & >
- class **promise**<\_Res >

#### 5.442.1 Detailed Description

`template<typename _Res>`  
`class std::future<_Res & >`

Partial specialization for `future<R&>`

#### 5.442.2 Constructor & Destructor Documentation

##### **future()** [1/2]

```
template<typename _Res>
std::future<_Res & >::future (
    future<_Res & > && __uf) [inline], [noexcept]
```

Move constructor.

##### **future()** [2/2]

```
std::future<_Res >::future (
    future<_Res & > && __uf) [inline], [noexcept]
```

Move constructor.

#### 5.442.3 Member Function Documentation

##### **\_M\_get\_result()**

```
__result_type std::__basic_future<_Res >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

##### **get()** [1/2]

```
_Res std::future<_Res >::get () [inline]
```

Retrieving the value.

##### **get()** [2/2]

```
template<typename _Res>
_Res & std::future<_Res & >::get () [inline]
```

Retrieving the value.

The documentation for this class was generated from the following file:

- [future](#)

#### 5.443 std::future< void > Class Reference

```
#include <future>
```

Inheritance diagram for std::future< void >:



### Public Member Functions

- **future** (const [future](#) &)=delete
- **future** (const [future](#) &)=delete
- **future** ([future](#) &&\_\_uf) noexcept
- **future** ([future](#) &&\_\_uf) noexcept
- void **get** ()
- void **get** ()
- **future** & **operator=** (const [future](#) &)=delete
- **future** & **operator=** (const [future](#) &)=delete
- **future** & **operator=** ([future](#) &&\_\_fut) noexcept
- **future** & **operator=** ([future](#) &&\_\_fut) noexcept
- **shared\_future**< void > **share** () noexcept
- **shared\_future**< void > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- **future\_status** **wait\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rel) const
- **future\_status** **wait\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs) const

### Protected Types

- typedef \_\_future\_base::\_Result< void > & \_\_**result\_type**

### Protected Member Functions

- \_\_result\_type **\_M\_get\_result** () const
- void **\_M\_swap** (\_\_[basic\\_future](#) &\_\_that) noexcept

### Friends

- **future**< \_\_async\_result\_of< \_Fn, \_Args... > > **async** ([launch](#), \_Fn &&, \_Args &&...)
- template<typename \_Fn, typename... \_Args>  
**future**< \_\_async\_result\_of< \_Fn, \_Args... > > **async** ([launch](#), \_Fn &&, \_Args &&...)
- class **packaged\_task**

- `template<typename>`  
class **packaged\_task**
- class **promise**< **\_Res** >
- class **promise**< **void** >

#### 5.443.1 Detailed Description

Explicit specialization for `future<void>`

#### 5.443.2 Constructor & Destructor Documentation

##### **future()** [1/2]

```
std::future< void >::future (
    future< void > && __uf) [inline], [noexcept]
```

Move constructor.

##### **future()** [2/2]

```
std::future< void >::future (
    future< void > && __uf) [inline], [noexcept]
```

Move constructor.

#### 5.443.3 Member Function Documentation

##### **\_M\_get\_result()**

```
__result_type std::__basic_future< void >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

##### **get()** [1/2]

```
void std::future< void >::get () [inline]
```

Retrieving the value.

##### **get()** [2/2]

```
void std::future< void >::get () [inline]
```

Retrieving the value.

The documentation for this class was generated from the following file:

- `future`

#### 5.444 `std::future_error` Class Reference

```
#include <future>
```

Inheritance diagram for `std::future_error`:



#### Public Member Functions

- `future_error` (`future_errc` `__errc`)
- `const error_code & code` () `const noexcept`
- `virtual const char * what` () `const noexcept`

#### Friends

- `void __throw_future_error` (int)

#### 5.444.1 Detailed Description

Exception type thrown by futures.

Since

C++11

#### 5.444.2 Member Function Documentation

##### `what()`

```
virtual const char * std::future_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::logic\\_error](#).

The documentation for this class was generated from the following file:

- [future](#)

## 5.445 `std::gamma_distribution<_RealType>` Class Template Reference

```
#include <random>
```

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [gamma\\_distribution](#) ()
- [gamma\\_distribution](#) (`_RealType __alpha_val, _RealType __beta_val= _RealType(1)`)
- [gamma\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void [\\_\\_generate](#) (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void [\\_\\_generate](#) (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- template<typename `_UniformRandomNumberGenerator`>  
void [\\_\\_generate](#) (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `_RealType alpha` () const
- `_RealType beta` () const
- `result_type max` () const
- `result_type min` () const
- template<typename `_UniformRandomNumberGenerator`>  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator`>  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- [param\\_type param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- template<typename `_RealType1`, typename `_CharT`, typename `_Traits`>  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::gamma\_distribution< _RealType1 > &__x)`
- bool `operator==` (const [gamma\\_distribution](#) &\_\_d1, const [gamma\\_distribution](#) &\_\_d2)
- template<typename `_RealType1`, typename `_CharT`, typename `_Traits`>  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::gamma\_distribution< _RealType1 > &__x)`

### 5.445.1 Detailed Description

```
template<typename _RealType = double>
class std::gamma_distribution< _RealType >
```

A gamma continuous distribution for random numbers.  
The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Since

C++11

### 5.445.2 Member Typedef Documentation

#### `result_type`

```
template<typename _RealType = double>
typedef _RealType std::gamma_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 5.445.3 Constructor & Destructor Documentation

#### `gamma_distribution()` [1/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution () [inline]
```

Constructs a gamma distribution with parameters 1 and 1.

#### `gamma_distribution()` [2/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution (
    _RealType __alpha_val,
    _RealType __beta_val = _RealType(1)) [inline], [explicit]
```

Constructs a gamma distribution with parameters  $\alpha$  and  $\beta$ .

### 5.445.4 Member Function Documentation

#### `alpha()`

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::alpha () const [inline]
```

Returns the  $\alpha$  of the distribution.

#### `beta()`

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::beta () const [inline]
```

Returns the  $\beta$  of the distribution.

#### `max()`

```
template<typename _RealType = double>
result_type std::gamma_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### `min()`

```
template<typename _RealType = double>
result_type std::gamma_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.



**operator>() [1/2]**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::gamma_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)](#).

**operator>() [2/2]**

```
template<typename _RealType>
template<typename _UniformRandomNumberGenerator>
gamma_distribution< _RealType >::result_type std::gamma_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng,
    const param_type & __param)
```

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables" ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

References [std::log\(\)](#), and [std::pow\(\)](#).

**param() [1/2]**

```
template<typename _RealType = double>
param_type std::gamma_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

**5.445.5 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::gamma_distribution< _RealType1 > & __x) [friend]
```

Inserts a `gamma_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <code>__os</code> | An output stream.                                             |
| <code>__x</code>  | A <code>gamma_distribution</code> random number distribution. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
    const gamma\_distribution< _RealType > & __d1,
    const gamma\_distribution< _RealType > & __d2) [friend]
```

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic\_istream< _CharT, _Traits > & operator>> (
    std::basic\_istream< _CharT, _Traits > & __is,
    std::gamma\_distribution< _RealType1 > & __x) [friend]
```

Extracts a `gamma_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                  |
| <code>__x</code>  | A <code>gamma_distribution</code> random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.446 `std::geometric_distribution<_IntType>` Class Template Reference**

```
#include <random>
```

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- **geometric\_distribution** (const [param\\_type](#) &\_\_p)
- **geometric\_distribution** (double \_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const [geometric\\_distribution](#) &\_\_d1, const [geometric\\_distribution](#) &\_\_d2)

### 5.446.1 Detailed Description

```
template<typename _IntType = int>
class std::geometric_distribution< _IntType >
```

A discrete geometric random number distribution.

The formula for the geometric probability density function is  $p(i|p) = p(1 - p)^i$  where  $p$  is the parameter of the distribution.

Since

C++11

### 5.446.2 Member Typedef Documentation

#### result\_type

```
template<typename _IntType = int>
typedef _IntType std::geometric\_distribution< _IntType >::result_type
```

The type of the range of the distribution.

### 5.446.3 Member Function Documentation

#### max()

```
template<typename _IntType = int>
result\_type std::geometric\_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

**min()**

```
template<typename _IntType = int>
result_type std::geometric_distribution<_IntType>::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

**operator()()**

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::geometric_distribution<_IntType>::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by `operator()()`.

**p()**

```
template<typename _IntType = int>
double std::geometric_distribution<_IntType>::p () const [inline]
```

Returns the distribution parameter `p`.

**param() [1/2]**

```
template<typename _IntType = int>
param_type std::geometric_distribution<_IntType>::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _IntType = int>
void std::geometric_distribution<_IntType>::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _IntType = int>
void std::geometric_distribution<_IntType>::reset () [inline]
```

Resets the distribution state.  
Does nothing for the geometric distribution.

**5.446.4 Friends And Related Symbol Documentation****operator==**

```
template<typename _IntType = int>
bool operator== (
    const geometric_distribution<_IntType> & __d1,
    const geometric_distribution<_IntType> & __d2) [friend]
```

Return true if two geometric distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 5.447 `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

#include <assoc\_container.hpp>

Inheritance diagram for `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`:



#### Public Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn**
- typedef [gp\\_hash\\_tag](#) **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef Probe\_Fn **probe\_fn**
- typedef Resize\_Policy **resize\_policy**

#### Public Member Functions

- [gp\\_hash\\_table](#) ()
- **gp\_hash\_table** (const [gp\\_hash\\_table](#) &other)
- [gp\\_hash\\_table](#) (const hash\_fn &h)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- template<typename It>  
[gp\\_hash\\_table](#) (It first, It last)
- template<typename It>  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h)
- template<typename It>  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e)
- template<typename It>  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- template<typename It>  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)

- `template<typename It>`  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- `gp_hash_table` & **operator=** (const `gp_hash_table` &other)
- void **swap** (`gp_hash_table` &other)

### 5.447.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A general-probing hash-based associative container.

#### Template Parameters

|                      |                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                        |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                        |
| <i>Hash_Fn</i>       | Hashing functor.                                                                                                                                                                                 |
| <i>Eq_Fn</i>         | Equal functor.                                                                                                                                                                                   |
| <i>Comb_Probe_Fn</i> | Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. |
| <i>Probe_Fn</i>      | Probe functor.                                                                                                                                                                                   |
| <i>Resize_Policy</i> | Resizes hash.                                                                                                                                                                                    |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true                                          |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                  |

Base tag choices are: `gp_hash_tag`.

Base is `basic_hash_table`.

### 5.447.2 Constructor & Destructor Documentation

#### `gp_hash_table()` [1/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table () [inline]
```

Default constructor.

#### `gp_hash_table()` [2/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
```

```

detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h) [inline]

```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

### **gp\_hash\_table()** [3/12]

```

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h,
    const eq_fn & e) [inline]

```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

### **gp\_hash\_table()** [4/12]

```

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp) [inline]

```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

### **gp\_hash\_table()** [5/12]

```

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp,
    const probe_fn & p) [inline]

```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

**gp\_hash\_table()** [6/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp,
    const probe_fn & p,
    const resize_policy & rp) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `Resize_Policy` object of the container object.

**gp\_hash\_table()** [7/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

**gp\_hash\_table()** [8/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
    const hash_fn & h) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

**gp\_hash\_table()** [9/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
```



```

detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e) [inline]

```

Constructor taking \_\_iterators to a range of value\_types and some policy objects. The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, and r\_eq\_fn will be copied by the eq\_fn object of the container object.

### gp\_hash\_table() [10/12]

```

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp) [inline]

```

Constructor taking \_\_iterators to a range of value\_types and some policy objects. The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, and r\_comb\_probe\_fn will be copied by the comb\_probe\_fn object of the container object.

### gp\_hash\_table() [11/12]

```

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp,
    const probe_fn & p) [inline]

```

Constructor taking \_\_iterators to a range of value\_types and some policy objects. The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, r\_comb\_probe\_fn will be copied by the comb\_probe\_fn object of the container object, and r\_probe\_fn will be copied by the probe\_fn object of the container object.

**gp\_hash\_table()** [12/12]

```

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp,
    const probe_fn & p,
    const resize_policy & rp) [inline]

```

Constructor taking \_\_iterators to a range of value\_types and some policy objects. The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, r\_comb\_probe\_fn will be copied by the comb\_ probe\_fn object of the container object, r\_probe\_fn will be copied by the probe\_fn object of the container object, and r\_resize\_policy will be copied by the resize\_policy object of the container object.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

**5.448 \_\_gnu\_pbds::gp\_hash\_tag Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::gp\_hash\_tag:



### 5.448.1 Detailed Description

General-probing hash.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.449 `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >` Class Template Reference

```
#include <gp_ht_map_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`:



### Public Types

- enum { **store\_hash** }
- typedef `_Alloc` **allocator\_type**
- typedef `Comb_Probe_Fn` **comb\_probe\_fn**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**

- typedef `point_const_iterator` **point\_const\_iterator**
- typedef `point_iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `Probe_Fn` **probe\_fn**
- typedef `traits_base::reference` **reference**
- typedef `Resize_Policy` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `traits_base::value_type` **value\_type**

## Public Member Functions

- **gp\_ht\_map** (const `gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &`)
- **gp\_ht\_map** (const `Hash_Fn &`)
- **gp\_ht\_map** (const `Hash_Fn &`, const `Eq_Fn &`)
- **gp\_ht\_map** (const `Hash_Fn &`, const `Eq_Fn &`, const `Comb_Probe_Fn &`)
- **gp\_ht\_map** (const `Hash_Fn &`, const `Eq_Fn &`, const `Comb_Probe_Fn &`, const `Probe_Fn &`)
- **gp\_ht\_map** (const `Hash_Fn &`, const `Eq_Fn &`, const `Comb_Probe_Fn &`, const `Probe_Fn &`, const `Resize_Policy &`)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It>  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- point\_iterator **find\_end** ()
- point\_const\_iterator **find\_end** () const
- Comb\_Probe\_Fn & **get\_comb\_probe\_fn** ()
- const Comb\_Probe\_Fn & **get\_comb\_probe\_fn** () const
- Eq\_Fn & **get\_eq\_fn** ()
- const Eq\_Fn & **get\_eq\_fn** () const
- Hash\_Fn & **get\_hash\_fn** ()
- const Hash\_Fn & **get\_hash\_fn** () const
- Probe\_Fn & **get\_probe\_fn** ()
- const Probe\_Fn & **get\_probe\_fn** () const
- Resize\_Policy & **get\_resize\_policy** ()
- const Resize\_Policy & **get\_resize\_policy** () const
- `std::pair< point_iterator, bool >` **insert** (const\_reference r\_val)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** (`gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &`)

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Friends

- class **const\_iterator\_**
- class **iterator\_**

### 5.449.1 Detailed Description

template<typename Key, typename Mapped, typename Hash\_Fn, typename Eq\_Fn, typename \_Alloc, bool Store\_Hash, typename Comb\_Probe\_Fn, typename Probe\_Fn, typename Resize\_Policy>  
class **\_\_gnu\_pbds::detail::gp\_ht\_map**< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >

A general-probing hash-based container.

#### Template Parameters

|                      |                                                                                                                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                                                                                |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                                                                                |
| <i>Hash_Fn</i>       | Hashing functor. Default is <code>__gnu_cxx::hash</code> .                                                                                                                                                                                               |
| <i>Eq_Fn</i>         | Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>                                                                                                                                                                                             |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                                                                          |
| <i>Store_Hash</i>    | If key type stores extra metadata. Defaults to false.                                                                                                                                                                                                    |
| <i>Comb_Probe_Fn</i> | Combining probe functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default <code>direct_mask_range_hashing</code> . |
| <i>Probe_Fn</i>      | Probe functor. Defaults to <code>linear_probe_fn</code> , also <code>quadratic_probe_fn</code> .                                                                                                                                                         |
| <i>Resize_Policy</i> | Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .                                                                                   |

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_probe_fn`, `detail::types_traits`. (Optional: `detail::debug_map_base`.)

### 5.449.2 Member Enumeration Documentation

#### anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
anonymous enum
```

Value stores hash, true or false.

### 5.449.3 Member Function Documentation

#### empty()

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
bool __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_
Fn, Probe_Fn, Resize_Policy >::empty () const [inline], [nodiscard]
True if size() == 0.
```

### **get\_comb\_probe\_fn()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ()
Return current comb_probe_fn.
```

### **get\_comb\_probe\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn () const
Return current const comb_probe_fn.
```

### **get\_eq\_fn()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ()
Return current eq_fn.
```

### **get\_eq\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn () const
Return current const eq_fn.
```

### **get\_hash\_fn()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ()
Return current hash_fn.
```

### **get\_hash\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn () const
Return current const hash_fn.
```

### **get\_probe\_fn()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ()
Return current probe_fn.
```

**get\_probe\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn () const
Return current const probe_fn.
```

**get\_resize\_policy()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ()
Return current resize_policy.
```

**get\_resize\_policy()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy () const
Return current const resize_policy.
```

The documentation for this class was generated from the following file:

- [gp\\_ht\\_map.hpp](#)

## 5.450 std::chrono::gps\_clock Class Reference

```
#include <chrono>
```

### Public Types

- using **duration**
- using **period**
- using **rep**
- using **time\_point**

### Static Public Member Functions

- template<typename \_Duration>  
static [gps\\_time](#)< [common\\_type\\_t](#)< \_Duration, [seconds](#) > > **from\_utc** (const [utc\\_time](#)< \_Duration > &\_\_t)
- static [time\\_point](#) **now** ()
- template<typename \_Duration>  
static [utc\\_time](#)< [common\\_type\\_t](#)< \_Duration, [seconds](#) > > **to\_utc** (const [gps\\_time](#)< \_Duration > &\_\_t)

### Static Public Attributes

- static constexpr bool **is\_steady**

#### 5.450.1 Detailed Description

A clock that measures GPS time.  
The epoch is 1980-01-06 00:00:00.

Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

## 5.451 `std::greater<_Tp>` Struct Template Reference

```
#include <std_function.h>
```

Inheritance diagram for `std::greater<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 5.451.1 Detailed Description

```
template<typename _Tp>
struct std::greater<_Tp>
```

One of the [comparison functors](#).

#### 5.451.2 Member Typedef Documentation

##### `first_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### `result_type`

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
result_type is the return type
```



### second\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.452 std::ranges::greater Struct Reference

```
#include <ranges_cmp.h>
```

### Public Types

- using **is\_transparent**

### Public Member Functions

- `template<typename _Tp, typename _Up>`  
requires `totally_ordered_with<_Tp, _Up>`  
`constexpr bool operator() (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval<_Up >())<std::declval<_Tp >()))`

#### 5.452.1 Detailed Description

ranges::greater function object type.

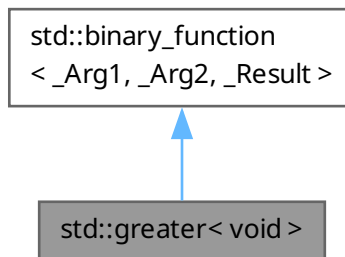
The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 5.453 std::greater< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::greater< void >:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` **is\_transparent**

- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)<\_Tp>(\_\_t) > [std::forward](#)<\_Up>(\_\_u))) -> decltype([std::forward](#)<\_Tp>(\_\_t) > [std::forward](#)<\_Up>(\_\_u))
- template<typename \_Tp, typename \_Up>  
constexpr bool **operator()** (\_Tp \*\_\_t, \_Up \*\_\_u) const noexcept
- constexpr bool **operator()** (const void &\_\_x, const void &\_\_y) const

#### 5.453.1 Detailed Description

One of the [comparison functors](#).

#### 5.453.2 Member Typedef Documentation

##### **first\_argument\_type**

typedef void [std::binary\\_function](#)< void, void, bool >::first\_argument\_type  
first\_argument\_type is the type of the first argument

##### **result\_type**

typedef bool [std::binary\\_function](#)< void, void, bool >::result\_type  
result\_type is the return type

##### **second\_argument\_type**

typedef void [std::binary\\_function](#)< void, void, bool >::second\_argument\_type  
second\_argument\_type is the type of the second argument  
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.454 `std::greater_equal<_Tp>` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::greater_equal<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 5.454.1 Detailed Description

```
template<typename _Tp>
struct std::greater_equal<_Tp>
```

One of the [comparison functors](#).

#### 5.454.2 Member Typedef Documentation

##### **first\_argument\_type**

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### **result\_type**

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
result_type is the return type
```

##### **second\_argument\_type**

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.455 std::ranges::greater\_equal Struct Reference

```
#include <ranges_cmp.h>
```

### Public Types

- using `is_transparent`

### Public Member Functions

- `template<typename _Tp, typename _Up>`  
`requires totally_ordered_with<_Tp, _Up>`  
`constexpr bool operator() (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval<_Tp>())<  
std::declval<_Up>()))`

#### 5.455.1 Detailed Description

`ranges::greater_equal` function object type.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 5.456 std::greater\_equal< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::greater_equal< void >`:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` `is_transparent`
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

## Public Member Functions

- `template<typename _Tp, typename _Up>`  
`constexpr auto operator() (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) >=std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) >=std::forward< _Up >(__u))`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator() (_Tp *__t, _Up *__u) const noexcept`
- `constexpr bool operator() (const void &__x, const void &__y) const`

### 5.456.1 Detailed Description

One of the [comparison functors](#).

### 5.456.2 Member Typedef Documentation

#### **first\_argument\_type**

`typedef void std::binary\_function< void, void, bool >::first_argument_type`  
`first_argument_type` is the type of the first argument

#### **result\_type**

`typedef bool std::binary\_function< void, void, bool >::result_type`  
`result_type` is the return type

#### **second\_argument\_type**

`typedef void std::binary\_function< void, void, bool >::second_argument_type`  
`second_argument_type` is the type of the second argument  
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.457 `__gnu_cxx::random_condition::group_adjustor` Struct Reference

```
#include <throw_allocator.h>
```

## Public Member Functions

- `group_adjustor (size_t size)`

### 5.457.1 Detailed Description

Group condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.458 `__gnu_parallel::growing_blocks_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for \_\_gnu\_parallel::growing\_blocks\_tag:



### 5.458.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_GROWING_BLOCKS`

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.459 std::gslice Class Reference

```
#include <gslice.h>
```

### Public Member Functions

- [gslice](#) ()
- [gslice](#) (const [gslice](#) &)
- [gslice](#) (size\_t \_\_o, const [valarray](#)< size\_t > &\_\_l, const [valarray](#)< size\_t > &\_\_s)
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- [valarray](#)< size\_t > [size](#) () const
- size\_t [start](#) () const
- [valarray](#)< size\_t > [stride](#) () const

### Friends

- template<typename \_Tp>  
class [valarray](#)

### 5.459.1 Detailed Description

Class defining multi-dimensional subset of an array.

The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size

and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size. For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

## 5.460 `std::gslice_array<_Tp>` Class Template Reference

```
#include <gslice_array.h>
```

### Public Types

- `typedef _Tp value_type`

### Public Member Functions

- [gslice\\_array](#) (const [gslice\\_array](#) &)
- `template<class _Dom>`  
void **operator%<sub>2D</sub>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator%<sub>2D</sub>** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator&<sub>2D</sub>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator&<sub>2D</sub>** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator\*<sub>2D</sub>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator\*<sub>2D</sub>** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator+<sub>2D</sub>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator+<sub>2D</sub>** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator-<sub>2D</sub>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator-<sub>2D</sub>** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator/<sub>2D</sub>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator/<sub>2D</sub>** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator<<<sub>2D</sub>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator<<<sub>2D</sub>** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator=** (const \_Tp &) const
- [gslice\\_array](#) & **operator=** (const [gslice\\_array](#) &)
- void **operator=** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator>>=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator>>=** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator^<sub>2D</sub>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator^<sub>2D</sub>** (const [valarray](#)< \_Tp > &) const
- `template<class _Dom>`  
void **operator|<sub>2D</sub>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator|<sub>2D</sub>** (const [valarray](#)< \_Tp > &) const

## Friends

- class **valarray**<\_Tp>

### 5.460.1 Detailed Description

**template**<typename \_Tp>  
**class** std::gslice\_array<\_Tp>

Reference to multi-dimensional subset of an array.

A gslice\_array is a reference to the actual elements of an array specified by a gslice. The way to get a gslice\_array is to call operator[](gslice) on a valarray. The returned gslice\_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this gslice\_array refers to.

#### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

### 5.460.2 Member Function Documentation

#### operator%=( )

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator%= (
    const valarray<_Tp> & ) const
```

Modulo slice elements by corresponding elements of *v*.

#### operator&=( )

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator&= (
    const valarray<_Tp> & ) const
```

Logical and slice elements with corresponding elements of *v*.

#### operator\*=( )

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator*= (
    const valarray<_Tp> & ) const
```

Multiply slice elements by corresponding elements of *v*.

#### operator+=( )

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator+= (
    const valarray<_Tp> & ) const
```

Add corresponding elements of *v* to slice elements.

#### operator-=( )

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator-= (
    const valarray<_Tp> & ) const
```

Subtract corresponding elements of *v* from slice elements.



**operator/=( )**

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator/= (
    const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

**operator<<=( )**

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator<<= (
    const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

**operator>>=( )**

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator>>= (
    const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.

References [gslice\\_array\(\)](#).

**operator^=( )**

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator^= (
    const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

**operator" |=()**

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator|= (
    const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following files:

- [valarray](#)
- [gslice\\_array.h](#)

**5.461 std::has\_virtual\_destructor< \_Tp > Struct Template Reference**

```
#include <type_traits>
```

**5.461.1 Detailed Description**

```
template<typename _Tp>
struct std::has_virtual_destructor< _Tp >
```

has\_virtual\_destructor

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.462 std::hash< \_Tp > Struct Template Reference**

```
#include <functional_hash.h>
```

#### 5.462.1 Detailed Description

```
template<typename _Tp>
struct std::hash<_Tp>
```

Primary class template hash.

Primary class template hash, usable for enum types only.

The documentation for this struct was generated from the following file:

- [string\\_view](#)

### 5.463 `std::hash<__debug::bitset<_Nb>>` Struct Template Reference

```
#include <bitset>
```

#### Public Member Functions

- `size_t operator() (const \_\_debug::bitset<\_Nb> &__b) const` noexcept

#### 5.463.1 Detailed Description

```
template<size_t _Nb>
struct std::hash<__debug::bitset<_Nb>>
```

`std::hash` specialization for `bitset`.

The documentation for this struct was generated from the following file:

- [debug/bitset](#)

### 5.464 `std::hash<__debug::vector<bool,_Alloc>>` Struct Template Reference

```
#include <vector>
```

#### Public Member Functions

- `size_t operator() (const \_\_debug::vector<bool,\_Alloc> &__b) const` noexcept

#### 5.464.1 Detailed Description

```
template<typename _Alloc>
struct std::hash<__debug::vector<bool,_Alloc>>
```

`std::hash` specialization for `vector<bool>`.

The documentation for this struct was generated from the following file:

- [debug/vector](#)

### 5.465 `std::hash<__gnu_cxx::__u16vstring>` Struct Reference

```
#include <vstring.h>
```

#### Public Member Functions

- `size_t operator() (const \_\_gnu\_cxx::\_\_u16vstring &__s) const` noexcept

#### 5.465.1 Detailed Description

std::hash specialization for \_\_u16vstring.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

#### 5.466 std::hash< \_\_gnu\_cxx::\_\_u32vstring > Struct Reference

```
#include <vstring.h>
```

##### Public Member Functions

- `size_t operator() (const __gnu_cxx::__u32vstring &__s) const` noexcept

##### 5.466.1 Detailed Description

std::hash specialization for \_\_u32vstring.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

#### 5.467 std::hash< \_\_gnu\_cxx::\_\_vstring > Struct Reference

```
#include <vstring.h>
```

##### Public Member Functions

- `size_t operator() (const __gnu_cxx::__vstring &__s) const` noexcept

##### 5.467.1 Detailed Description

std::hash specialization for \_\_vstring.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

#### 5.468 std::hash< \_\_gnu\_cxx::\_\_wvstring > Struct Reference

```
#include <vstring.h>
```

##### Public Member Functions

- `size_t operator() (const __gnu_cxx::__wvstring &__s) const` noexcept

##### 5.468.1 Detailed Description

std::hash specialization for \_\_wvstring.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

**5.469 `std::hash< __gnu_cxx::throw_value_limit >` Struct Reference**

```
#include <throw_allocator.h>
```

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_limit >`:

**Public Types**

- typedef `__gnu_cxx::throw_value_limit` `argument_type`
- typedef `size_t` `result_type`

**Public Member Functions**

- `size_t operator()` (const `__gnu_cxx::throw_value_limit` &\_\_val) const

**5.469.1 Detailed Description**

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

**5.469.2 Member Typedef Documentation****`argument_type`**

```
typedef __gnu_cxx::throw_value_limit std::unary_function< __gnu_cxx::throw_value_limit, size_t >::argument_type [inherited]
```

`argument_type` is the type of the argument

**`result_type`**

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_limit, size_t >::result_type [inherited]
```

`result_type` is the return type

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.470 std::hash< \_\_gnu\_cxx::throw\_value\_random > Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for std::hash< \_\_gnu\_cxx::throw\_value\_random >:



### Public Types

- typedef `__gnu_cxx::throw_value_random` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator()` (const `__gnu_cxx::throw_value_random` &`__val`) const

#### 5.470.1 Detailed Description

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_random`.

#### 5.470.2 Member Typedef Documentation

##### `argument_type`

```
typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random, size_t >::argument_type [inherited]
```

`argument_type` is the type of the argument

##### `result_type`

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_random, size_t >::result_type [inherited]
```

`result_type` is the return type

The documentation for this struct was generated from the following file:

- `throw_allocator.h`

**5.471 `std::hash< __gnu_debug::basic_string< _CharT > >` Struct Template Reference**

```
#include <string>
```

Inheritance diagram for `std::hash< __gnu_debug::basic_string< _CharT > >`:

**5.471.1 Detailed Description**

```
template<typename _CharT>
```

```
struct std::hash< __gnu_debug::basic_string< _CharT > >
```

`std::hash` specialization for `__gnu_debug::basic_string`.

The documentation for this struct was generated from the following file:

- [debug/string](#)

**5.472 `std::hash< __shared_ptr< _Tp, _Lp > >` Struct Template Reference**

```
#include <shared_ptr_base.h>
```

**Public Member Functions**

- `size_t operator() (const __shared_ptr< _Tp, _Lp > &__s) const` noexcept

**5.472.1 Detailed Description**

```
template<typename _Tp, _Lock_policy _Lp>
```

```
struct std::hash< __shared_ptr< _Tp, _Lp > >
```

`std::hash` specialization for `__shared_ptr`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

**5.473 `std::hash< _Tp * >` Struct Template Reference**

```
#include <functional_hash.h>
```

**Public Member Functions**

- `size_t operator() (_Tp *__p) const` noexcept

#### 5.473.1 Detailed Description

```
template<typename _Tp>
struct std::hash<_Tp*>
```

Partial specializations for pointer types.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 5.474 std::hash< basic\_string< char, char\_traits< char >, \_Alloc > > Struct Template Reference

```
#include <basic_string.h>
```

##### Public Member Functions

- `size_t operator() (const _StrT &__s) const noexcept`

#### 5.474.1 Detailed Description

```
template<typename _Alloc>
struct std::hash< basic_string< char, char_traits< char >, _Alloc > >
```

std::hash specialization for string.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

#### 5.475 std::hash< basic\_string< char16\_t, char\_traits< char16\_t >, \_Alloc > > Struct Template Reference

```
#include <basic_string.h>
```

##### Public Member Functions

- `size_t operator() (const _StrT &__s) const noexcept`

#### 5.475.1 Detailed Description

```
template<typename _Alloc>
struct std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >
```

std::hash specialization for u16string.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

#### 5.476 std::hash< basic\_string< char32\_t, char\_traits< char32\_t >, \_Alloc > > Struct Template Reference

```
#include <basic_string.h>
```

##### Public Member Functions

- `size_t operator() (const _StrT &__s) const noexcept`

#### 5.476.1 Detailed Description

`template<typename _Alloc>`

`struct std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >`

`std::hash` specialization for `u32string`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

### 5.477 `std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >` Struct Template Reference

`#include <basic_string.h>`

#### Public Member Functions

- `size_t operator() (const _StrT &__s) const` noexcept

#### 5.477.1 Detailed Description

`template<typename _Alloc>`

`struct std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >`

`std::hash` specialization for `wstring`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

### 5.478 `std::hash< bool >` Struct Reference

`#include <functional_hash.h>`

#### Public Member Functions

- `size_t operator() (bool __val) const` noexcept

#### 5.478.1 Detailed Description

Explicit specialization for `bool`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.479 `std::hash< char >` Struct Reference

`#include <functional_hash.h>`

#### Public Member Functions

- `size_t operator() (char __val) const` noexcept

#### 5.479.1 Detailed Description

Explicit specialization for `char`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)



### 5.480 `std::hash< char16_t >` Struct Reference

```
#include <functional_hash.h>
```

#### Public Member Functions

- `size_t operator()` (`char16_t __val`) `const noexcept`

#### 5.480.1 Detailed Description

Explicit specialization for `char16_t`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.481 `std::hash< char32_t >` Struct Reference

```
#include <functional_hash.h>
```

#### Public Member Functions

- `size_t operator()` (`char32_t __val`) `const noexcept`

#### 5.481.1 Detailed Description

Explicit specialization for `char32_t`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.482 `std::hash< double >` Struct Reference

```
#include <functional_hash.h>
```

#### Public Member Functions

- `size_t operator()` (`double __val`) `const noexcept`

#### 5.482.1 Detailed Description

Specialization for `double`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.483 `std::hash< error_code >` Struct Reference

```
#include <system_error>
```

#### Public Member Functions

- `size_t operator()` (`const error\_code &__e`) `const noexcept`

#### 5.483.1 Detailed Description

`std::hash` specialization for `error_code`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.484 `std::hash< error_condition >` Struct Reference

```
#include <system_error>
```

### Public Member Functions

- `size_t operator()` (const [error\\_condition](#) &\_\_e) const noexcept

#### 5.484.1 Detailed Description

`std::hash` specialization for `error_condition`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.485 `std::hash< experimental::optional< _Tp > >` Struct Template Reference

```
#include <optional>
```

### Public Types

- using `argument_type`
- using `result_type`

### Public Member Functions

- `size_t operator()` (const [experimental::optional](#)< \_Tp > &\_\_t) const noexcept(noexcept([hash](#)< \_Tp > {}(\*\_\_t)))

#### 5.485.1 Detailed Description

```
template<typename _Tp>
```

```
struct std::hash< experimental::optional< _Tp > >
```

`std::hash` partial specialization for `experimental::optional`

The documentation for this struct was generated from the following file:

- [experimental/optional](#)

## 5.486 `std::hash< experimental::shared_ptr< _Tp > >` Struct Template Reference

```
#include <shared_ptr.h>
```

### Public Member Functions

- `size_t operator()` (const `experimental::shared_ptr`< \_Tp > &\_\_s) const noexcept

#### 5.486.1 Detailed Description

```
template<typename _Tp>
```

```
struct std::hash< experimental::shared_ptr< _Tp > >
```

`std::hash` specialization for `shared_ptr`.

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

### 5.487 `std::hash< float >` Struct Reference

```
#include <functional_hash.h>
```

#### Public Member Functions

- `size_t operator()` (float \_\_val) const noexcept

#### 5.487.1 Detailed Description

Specialization for float.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.488 `std::hash< int >` Struct Reference

```
#include <functional_hash.h>
```

#### Public Member Functions

- `size_t operator()` (int \_\_val) const noexcept

#### 5.488.1 Detailed Description

Explicit specialization for int.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.489 `std::hash< long >` Struct Reference

```
#include <functional_hash.h>
```

#### Public Member Functions

- `size_t operator()` (long \_\_val) const noexcept

#### 5.489.1 Detailed Description

Explicit specialization for long.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.490 `std::hash< long double >` Struct Reference

```
#include <functional_hash.h>
```

#### Public Member Functions

- `size_t operator()` (long double \_\_val) const noexcept

#### 5.490.1 Detailed Description

Specialization for long double.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.491 `std::hash< long long >` Struct Reference

```
#include <functional_hash.h>
```

### Public Member Functions

- `size_t operator() (long long __val) const` noexcept

#### 5.491.1 Detailed Description

Explicit specialization for `long long`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.492 `std::hash< shared_ptr< _Tp > >` Struct Template Reference

```
#include <shared_ptr.h>
```

### Public Member Functions

- `size_t operator() (const shared\_ptr< \_Tp > &__s) const` noexcept

#### 5.492.1 Detailed Description

```
template<typename _Tp>  
struct std::hash< shared_ptr< _Tp > >
```

`std::hash` specialization for `shared_ptr`.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 5.493 `std::hash< short >` Struct Reference

```
#include <functional_hash.h>
```

### Public Member Functions

- `size_t operator() (short __val) const` noexcept

#### 5.493.1 Detailed Description

Explicit specialization for `short`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.494 `std::hash< signed char >` Struct Reference

```
#include <functional_hash.h>
```

### Public Member Functions

- `size_t operator() (signed char __val) const` noexcept

#### 5.494.1 Detailed Description

Explicit specialization for signed char.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 5.495 std::hash< thread::id > Struct Reference

```
#include <std_thread.h>
```

##### Public Member Functions

- `size_t operator()` (const [thread::id](#) &\_\_id) const noexcept

#### 5.495.1 Detailed Description

std::hash specialization for thread::id.

The documentation for this struct was generated from the following file:

- [std\\_thread.h](#)

#### 5.496 std::hash< type\_index > Struct Reference

```
#include <typeindex>
```

##### Public Types

- typedef [type\\_index](#) `argument_type`
- typedef `size_t` `result_type`

##### Public Member Functions

- `size_t operator()` (const [type\\_index](#) &\_\_ti) const noexcept

#### 5.496.1 Detailed Description

std::hash specialization for type\_index.

The documentation for this struct was generated from the following file:

- [typeindex](#)

#### 5.497 std::hash< unique\_ptr< \_Tp, \_Dp > > Struct Template Reference

```
#include <unique_ptr.h>
```

#### 5.497.1 Detailed Description

```
template<typename _Tp, typename _Dp>  
struct std::hash< unique_ptr< _Tp, _Dp > >
```

std::hash specialization for unique\_ptr.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 5.498 `std::hash< unsigned char >` Struct Reference

```
#include <functional_hash.h>
```

### Public Member Functions

- `size_t operator() (unsigned char __val) const` noexcept

#### 5.498.1 Detailed Description

Explicit specialization for unsigned char.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.499 `std::hash< unsigned int >` Struct Reference

```
#include <functional_hash.h>
```

### Public Member Functions

- `size_t operator() (unsigned int __val) const` noexcept

#### 5.499.1 Detailed Description

Explicit specialization for unsigned int.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.500 `std::hash< unsigned long >` Struct Reference

```
#include <functional_hash.h>
```

### Public Member Functions

- `size_t operator() (unsigned long __val) const` noexcept

#### 5.500.1 Detailed Description

Explicit specialization for unsigned long.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.501 `std::hash< unsigned long long >` Struct Reference

```
#include <functional_hash.h>
```

### Public Member Functions

- `size_t operator() (unsigned long long __val) const` noexcept

#### 5.501.1 Detailed Description

Explicit specialization for unsigned long long.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.502 `std::hash< unsigned short >` Struct Reference

```
#include <functional_hash.h>
```

#### Public Member Functions

- `size_t operator() (unsigned short __val) const` noexcept

#### 5.502.1 Detailed Description

Explicit specialization for unsigned short.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.503 `std::hash< wchar_t >` Struct Reference

```
#include <functional_hash.h>
```

#### Public Member Functions

- `size_t operator() (wchar_t __val) const` noexcept

#### 5.503.1 Detailed Description

Explicit specialization for wchar\_t.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.504 `std::hash<::bitset< _Nb > >` Struct Template Reference

```
#include <bitset>
```

#### Public Member Functions

- `size_t operator() (const ::bitset< _Nb > &__b) const` noexcept

#### 5.504.1 Detailed Description

`template<size_t _Nb>`

`struct std::hash<::bitset< _Nb > >`

`std::hash` specialization for `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

### 5.505 `std::hash<::vector< bool, _Alloc > >` Struct Template Reference

```
#include <stl_bvector.h>
```

#### Public Member Functions

- `size_t operator() (const ::vector< bool, _Alloc > &) const` noexcept

### 5.505.1 Detailed Description

```
template<typename _Alloc>
struct std::hash<::vector< bool, _Alloc > >
```

std::hash specialization for vector<bool>.

The documentation for this struct was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

## 5.506 \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, Store\_Hash > Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, Store\_Hash >:



### 5.506.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >
```

Primary template.

The documentation for this struct was generated from the following file:



- [hash\\_eq\\_fn.hpp](#)

### 5.507 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >` Struct Template Reference

```
#include <hash_eq_fn.hpp>
```

#### Public Types

- typedef Eq\_Fn **eq\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**

#### Public Member Functions

- **hash\_eq\_fn** (const Eq\_Fn &r\_eq\_fn)
- bool **operator()** (key\_const\_reference r\_lhs\_key, key\_const\_reference r\_rhs\_key) const
- void **swap** (const [hash\\_eq\\_fn](#) &other)

#### 5.507.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc>  
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >
```

Specialization 1 - The client requests that hash values not be stored.  
The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

### 5.508 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >` Struct Template Reference

```
#include <hash_eq_fn.hpp>
```

#### Public Types

- typedef Eq\_Fn **eq\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef \_Alloc::size\_type **size\_type**

#### Public Member Functions

- **hash\_eq\_fn** (const Eq\_Fn &r\_eq\_fn)
- bool **operator()** (key\_const\_reference r\_lhs\_key, size\_type lhs\_hash, key\_const\_reference r\_rhs\_key, size\_type rhs\_hash) const
- void **swap** (const [hash\\_eq\\_fn](#) &other)

#### 5.508.1 Detailed Description

```
template<typename Key, class Eq_Fn, class _Alloc>  
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >
```

Specialization 2 - The client requests that hash values be stored.  
The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

## 5.509 `__gnu_pbds::hash_exponential_size_policy< Size_Type >` Class Template Reference

`#include <hash_policy.hpp>`

Inheritance diagram for `__gnu_pbds::hash_exponential_size_policy< Size_Type >`:



### Public Types

- typedef `Size_Type` **size\_type**

### Public Member Functions

- [hash\\_exponential\\_size\\_policy](#) (`size_type start_size=8`, `size_type grow_factor=2`)
- void **swap** ([hash\\_exponential\\_size\\_policy< Size\\_Type >](#) &other)

### Protected Member Functions

- `size_type` **get\_nearest\_larger\_size** (`size_type size`) const
- `size_type` **get\_nearest\_smaller\_size** (`size_type size`) const

#### 5.509.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::hash_exponential_size_policy< Size_Type >
```

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2).

#### 5.509.2 Constructor & Destructor Documentation

##### **hash\_exponential\_size\_policy()**

```
template<typename Size_Type>
__gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy (
    size_type start_size = 8,
    size_type grow_factor = 2)
```

Default constructor, or onstructor taking a `start_size`, or constructor taking a `start_size` and `grow_factor`. The policy will use the sequence of sizes `start_size`, `start_size* grow_factor`, `start_size* grow_factor^2`, ...

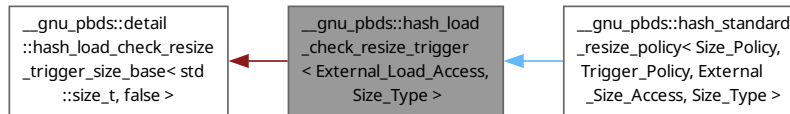
The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 5.510 `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`:



### Public Types

- enum { `external_load_access` }
- typedef `Size_Type` **size\_type**

### Public Member Functions

- `hash_load_check_resize_trigger` (float load\_min=0.125, float load\_max=0.5)
- `std::pair< float, float > get_loads ()` const
- void `set_loads` (`std::pair< float, float > load_pair`)
- void `swap` (`hash_load_check_resize_trigger` &other)

### Protected Member Functions

- bool `is_grow_needed` (size\_type size, size\_type num\_entries) const
- bool `is_resize_needed` () const
- void `notify_cleared` ()
- void `notify_erase_search_collision` ()
- void `notify_erase_search_end` ()
- void `notify_erase_search_start` ()
- void `notify_erased` (size\_type num\_entries)
- void `notify_externally_resized` (size\_type new\_size)
- void `notify_find_search_collision` ()
- void `notify_find_search_end` ()
- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (size\_type num\_entries)
- void `notify_resized` (size\_type new\_size)

#### 5.510.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on a load check. It keeps the load factor between some load factors `load_min` and `load_max`.

## 5.510.2 Member Enumeration Documentation

### anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

#### Enumerator

|                                   |                                                                                                                                                                 |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>external_load_access</code> | Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation. |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 5.510.3 Constructor & Destructor Documentation

### `hash_load_check_resize_trigger()`

```
template<bool External_Load_Access, typename Size_Type>
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_↵
resize_trigger (
    float load_min = 0.125,
    float load_max = 0.5)
```

Default constructor, or constructor taking `load_min` and `load_max` load factors between which this policy will keep the actual load.

## 5.510.4 Member Function Documentation

### `get_loads()`

```
template<bool External_Load_Access, typename Size_Type>
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_↵
_Type >::get_loads () const [inline]
```

Returns a pair of the minimal and maximal loads, respectively.

### `notify_cleared()`

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
cleared () [protected]
```

Notifies the table was cleared.

### `notify_inserted()`

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
inserted (
    size_type num_entries) [inline], [protected]
```

Notifies an element was inserted. The total number of entries in the table is `num_entries`.

### `notify_resized()`

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
resized (
    size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

References [notify\\_resized\(\)](#).  
 Referenced by [notify\\_resized\(\)](#).

### set\_loads()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
    std::pair< float, float > load_pair)
```

Sets the loads through a pair of the minimal and maximal loads, respectively.  
 The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 5.511 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, Hold\_Size > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, Hold\_Size >:



### 5.511.1 Detailed Description

```
template<typename Size_Type, bool Hold_Size>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >
```

Primary template.

The documentation for this class was generated from the following file:

- [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#)

## 5.512 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, true > Class Template Reference

```
#include <hash_load_check_resize_trigger_size_base.hpp>
```

## Protected Types

- typedef `Size_Type` **size\_type**

## Protected Member Functions

- `size_type` **get\_size** () const
- void **set\_size** (size\_type size)
- void **swap** ([hash\\_load\\_check\\_resize\\_trigger\\_size\\_base](#) &other)

### 5.512.1 Detailed Description

template<typename `Size_Type`>

class `__gnu_pbds::detail::hash_load_check_resize_trigger_size_base`< `Size_Type`, true >

Specializations.

The documentation for this class was generated from the following file:

- [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#)

## 5.513 `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >` Class Template Reference

```
#include <hash_map>
```

## Public Types

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**
- typedef `_Ht::const_pointer` **const\_pointer**
- typedef `_Ht::const_reference` **const\_reference**
- typedef `_Tp` **data\_type**
- typedef `_Ht::difference_type` **difference\_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::iterator` **iterator**
- typedef `_Ht::key_equal` **key\_equal**
- typedef `_Ht::key_type` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Ht::pointer` **pointer**
- typedef `_Ht::reference` **reference**
- typedef `_Ht::size_type` **size\_type**
- typedef `_Ht::value_type` **value\_type**

## Public Member Functions

- template<class `_InputIterator`>  
**hash\_map** (`_InputIterator` \_\_f, `_InputIterator` \_\_l)
- template<class `_InputIterator`>  
**hash\_map** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, `size_type` \_\_n)
- template<class `_InputIterator`>  
**hash\_map** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, `size_type` \_\_n, const hasher &\_\_hf)
- template<class `_InputIterator`>  
**hash\_map** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, `size_type` \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())

- **hash\_map** (size\_type \_\_n)
- **hash\_map** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_map** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator<\_\_type>())
- iterator **begin** ()
- const\_iterator **begin** () const
- size\_type **bucket\_count** () const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- void **erase** (iterator \_\_f, iterator \_\_l)
- void **erase** (iterator \_\_it)
- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_funct** () const
- template<class \_InputIterator>  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- pair< iterator, bool > **insert** (const value\_type &\_\_obj)
- pair< iterator, bool > **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- \_Tp & **operator[]** (const key\_type &\_\_key)
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** (hash\_map &\_\_hs)

## Friends

- template<class \_K1, class \_T1, class \_HF, class \_EqK, class \_AI>  
bool **operator==** (const hash\_map< \_K1, \_T1, \_HF, \_EqK, \_AI > &, const hash\_map< \_K1, \_T1, \_HF, \_EqK, \_AI > &)

### 5.513.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class
_Alloc = allocator<_Tp>>
class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

The documentation for this class was generated from the following file:

- [hash\\_map](#)

## 5.514 `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>` Class Template Reference

```
#include <hash_map>
```

### Public Types

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**
- typedef `_Ht::const_pointer` **const\_pointer**
- typedef `_Ht::const_reference` **const\_reference**
- typedef `_Tp` **data\_type**
- typedef `_Ht::difference_type` **difference\_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::iterator` **iterator**
- typedef `_Ht::key_equal` **key\_equal**
- typedef `_Ht::key_type` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Ht::pointer` **pointer**
- typedef `_Ht::reference` **reference**
- typedef `_Ht::size_type` **size\_type**
- typedef `_Ht::value_type` **value\_type**

### Public Member Functions

- template<class `_InputIterator`>  
**hash\_multimap** (`_InputIterator __f, _InputIterator __l`)
- template<class `_InputIterator`>  
**hash\_multimap** (`_InputIterator __f, _InputIterator __l, size_type __n`)
- template<class `_InputIterator`>  
**hash\_multimap** (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf`)
- template<class `_InputIterator`>  
**hash\_multimap** (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- **hash\_multimap** (`size_type __n`)
- **hash\_multimap** (`size_type __n, const hasher &__hf`)
- **hash\_multimap** (`size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- iterator **begin** ()
- const\_iterator **begin** () const
- size\_type **bucket\_count** () const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- void **erase** (iterator \_\_f, iterator \_\_l)
- void **erase** (iterator \_\_it)



- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_funct** () const
- template<class \_\_InputIterator>  
void **insert** (\_\_InputIterator \_\_f, \_\_InputIterator \_\_l)
- iterator **insert** (const value\_type &\_\_obj)
- iterator **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** (hash\_multimap &\_\_hs)

### Friends

- template<class \_K1, class \_T1, class \_HF, class \_EqK, class \_Al>  
bool **operator==** (const hash\_multimap< \_K1, \_T1, \_HF, \_EqK, \_Al > &, const hash\_multimap< \_K1, \_T1, \_HF, \_EqK, \_Al > &)

### 5.514.1 Detailed Description

template<class \_Key, class \_Tp, class \_HashFn = hash<\_Key>, class \_EqualKey = equal\_to<\_Key>, class \_Alloc = allocator<\_Tp>>  
class \_\_gnu\_cxx::hash\_multimap< \_Key, \_Tp, \_HashFn, \_EqualKey, \_Alloc >

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-<\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-<_style.html)

The documentation for this class was generated from the following file:

- [hash\\_map](#)

### 5.515 \_\_gnu\_cxx::hash\_multiset< \_Value, \_HashFcn, \_EqualKey, \_Alloc > Class Template Reference

```
#include <hash_set>
```

### Public Types

- typedef \_Ht::allocator\_type **allocator\_type**
- typedef \_Ht::const\_iterator **const\_iterator**
- typedef \_Alloc::const\_pointer **const\_pointer**
- typedef \_Alloc::const\_reference **const\_reference**
- typedef \_Ht::difference\_type **difference\_type**
- typedef \_Ht::hasher **hasher**
- typedef \_Ht::const\_iterator **iterator**
- typedef \_Ht::key\_equal **key\_equal**
- typedef \_Ht::key\_type **key\_type**
- typedef \_Alloc::pointer **pointer**
- typedef \_Alloc::reference **reference**
- typedef \_Ht::size\_type **size\_type**
- typedef \_Ht::value\_type **value\_type**

## Public Member Functions

- `template<class _InputIterator>`  
**hash\_multiset** (`_InputIterator __f, _InputIterator __l`)
- `template<class _InputIterator>`  
**hash\_multiset** (`_InputIterator __f, _InputIterator __l, size_type __n`)
- `template<class _InputIterator>`  
**hash\_multiset** (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf`)
- `template<class _InputIterator>`  
**hash\_multiset** (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- **hash\_multiset** (`size_type __n`)
- **hash\_multiset** (`size_type __n, const hasher &__hf`)
- **hash\_multiset** (`size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- iterator **begin** () const
- `size_type` **bucket\_count** () const
- void **clear** ()
- `size_type` **count** (`const key_type &__key`) const
- `size_type` **elems\_in\_bucket** (`size_type __n`) const
- bool **empty** () const
- iterator **end** () const
- `pair< iterator, iterator >` **equal\_range** (`const key_type &__key`) const
- `size_type` **erase** (`const key_type &__key`)
- void **erase** (`iterator __f, iterator __l`)
- void **erase** (`iterator __it`)
- iterator **find** (`const key_type &__key`) const
- `allocator_type` **get\_allocator** () const
- hasher **hash\_funct** () const
- `template<class _InputIterator>`  
void **insert** (`_InputIterator __f, _InputIterator __l`)
- iterator **insert** (`const value_type &__obj`)
- iterator **insert\_noresize** (`const value_type &__obj`)
- `key_equal` **key\_eq** () const
- `size_type` **max\_bucket\_count** () const
- `size_type` **max\_size** () const
- void **resize** (`size_type __hint`)
- `size_type` **size** () const
- void **swap** (`hash_multiset &hs`)

## Friends

- `template<class _Val, class _HF, class _EqK, class _Al>`  
bool **operator==** (`const hash_multiset<_Val, _HF, _EqK, _Al> &, const hash_multiset<_Val, _HF, _EqK, _Al> &`)

### 5.515.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc
= allocator<_Value>>
class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-<\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-<_style.html)

The documentation for this class was generated from the following file:

- [hash\\_set](#)

## 5.516 \_\_gnu\_pbds::hash\_prime\_size\_policy Class Reference

```
#include <hash_policy.hpp>
```

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [hash\\_prime\\_size\\_policy](#) ([size\\_type](#) start\_size=8)
- void **swap** ([hash\\_prime\\_size\\_policy](#) &other)

### Protected Member Functions

- [size\\_type](#) **get\_nearest\_larger\_size** ([size\\_type](#) size) const
- [size\\_type](#) **get\_nearest\_smaller\_size** ([size\\_type](#) size) const

#### 5.516.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

#### 5.516.2 Member Typedef Documentation

##### **size\_type**

```
typedef std::size_t __gnu_pbds::hash_prime_size_policy::size_type
```

Size type.

#### 5.516.3 Constructor & Destructor Documentation

##### **hash\_prime\_size\_policy()**

```
__gnu_pbds::hash_prime_size_policy::hash_prime_size_policy (  
    size\_type start_size = 8) [inline]
```

Default constructor, or onstructor taking a start\_size The policy will use the sequence of sizes approximately start\_size, start\_size\* 2, start\_size\* 2^2, ...

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 5.517 \_\_gnu\_cxx::hash\_set<\_Value, \_HashFcn, \_EqualKey, \_Alloc > Class Template Reference

```
#include <hash_set>
```

## Public Types

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `_Ht::difference_type` **difference\_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::const_iterator` **iterator**
- typedef `_Ht::key_equal` **key\_equal**
- typedef `_Ht::key_type` **key\_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Ht::size_type` **size\_type**
- typedef `_Ht::value_type` **value\_type**

## Public Member Functions

- template<class `_InputIterator`>  
**hash\_set** (`_InputIterator __f, _InputIterator __l`)
- template<class `_InputIterator`>  
**hash\_set** (`_InputIterator __f, _InputIterator __l, size_type __n`)
- template<class `_InputIterator`>  
**hash\_set** (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf`)
- template<class `_InputIterator`>  
**hash\_set** (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- **hash\_set** (`size_type __n`)
- **hash\_set** (`size_type __n, const hasher &__hf`)
- **hash\_set** (`size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- iterator **begin** () const
- `size_type` **bucket\_count** () const
- void **clear** ()
- `size_type` **count** (`const key_type &__key`) const
- `size_type` **elems\_in\_bucket** (`size_type __n`) const
- bool **empty** () const
- iterator **end** () const
- `pair< iterator, iterator >` **equal\_range** (`const key_type &__key`) const
- `size_type` **erase** (`const key_type &__key`)
- void **erase** (`iterator __f, iterator __l`)
- void **erase** (`iterator __it`)
- iterator **find** (`const key_type &__key`) const
- `allocator_type` **get\_allocator** () const
- hasher **hash\_func** () const
- template<class `_InputIterator`>  
void **insert** (`_InputIterator __f, _InputIterator __l`)
- `pair< iterator, bool >` **insert** (`const value_type &__obj`)
- `pair< iterator, bool >` **insert\_noresize** (`const value_type &__obj`)
- `key_equal` **key\_eq** () const
- `size_type` **max\_bucket\_count** () const
- `size_type` **max\_size** () const
- void **resize** (`size_type __hint`)
- `size_type` **size** () const
- void **swap** (`hash_set &__hs`)

## Friends

- `template<class _Val, class _HF, class _EqK, class _Al>`  
`bool operator==(const hash\_set< _Val, _HF, _EqK, _Al > &, const hash\_set< _Val, _HF, _EqK, _Al > &)`

### 5.517.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc
= allocator<_Value>>
class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

The documentation for this class was generated from the following file:

- [hash\\_set](#)

### 5.518 \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >`:



## Public Types

- enum { [external\\_load\\_access](#) }
- enum { [external\\_size\\_access](#) }
- typedef Size\_Policy [size\\_policy](#)
- typedef Size\_Type [size\\_type](#)
- typedef Trigger\_Policy [trigger\\_policy](#)

## Public Member Functions

- [hash\\_standard\\_resize\\_policy](#) ()
- [hash\\_standard\\_resize\\_policy](#) (const Size\_Policy &r\_size\_policy)
- [hash\\_standard\\_resize\\_policy](#) (const Size\_Policy &r\_size\_policy, const Trigger\_Policy &r\_trigger\_policy)
- size\_type [get\\_actual\\_size](#) () const
- `std::pair< float, float >` [get\\_loads](#) () const
- Size\_Policy & [get\\_size\\_policy](#) ()

- `const Size_Policy & get_size_policy () const`
- `Trigger_Policy & get_trigger_policy ()`
- `const Trigger_Policy & get_trigger_policy () const`
- `void resize (size_type suggested_new_size)`
- `void set_loads (std::pair< float, float > load_pair)`
- `void swap (hash_exponential_size_policy< Size_Type > &other)`
- `void swap (hash_load_check_resize_trigger &other)`
- `void swap (hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > &other)`

### Protected Member Functions

- `size_type get_nearest_larger_size (size_type size) const`
- `size_type get_nearest_smaller_size (size_type size) const`
- `size_type get_new_size (size_type size, size_type num_used_e) const`
- `bool is_grow_needed (size_type size, size_type num_entries) const`
- `bool is_resize_needed () const`
- `void notify_cleared ()`
- `void notify_erase_search_collision ()`
- `void notify_erase_search_end ()`
- `void notify_erase_search_start ()`
- `void notify_erased (size_type num_e)`
- `void notify_externally_resized (size_type new_size)`
- `void notify_find_search_collision ()`
- `void notify_find_search_end ()`
- `void notify_find_search_start ()`
- `void notify_insert_search_collision ()`
- `void notify_insert_search_end ()`
- `void notify_insert_search_start ()`
- `void notify_inserted (size_type num_e)`
- `void notify_resized (size_type new_size)`

#### 5.518.1 Detailed Description

```
template<typename Size_Policy = hash_exponential_size_policy<>, typename Trigger_Policy = hash_load_
check_resize_trigger<>, bool External_Size_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_
Type >
```

A resize policy which delegates operations to size and trigger policies.

#### 5.518.2 Member Enumeration Documentation

##### anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum [inherited]
```

##### Enumerator

|                                   |                                                                                                                                                                 |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>external_load_access</code> | Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation. |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|

### 5.518.3 Constructor & Destructor Documentation

#### hash\_standard\_resize\_policy() [1/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↵
_Type >::hash_standard_resize_policy ()
```

Default constructor.

#### hash\_standard\_resize\_policy() [2/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↵
_Type >::hash_standard_resize_policy (
    const Size_Policy & r_size_policy)
```

constructor taking some policies r\_size\_policy will be copied by the Size\_Policy object of this object.

#### hash\_standard\_resize\_policy() [3/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↵
_Type >::hash_standard_resize_policy (
    const Size_Policy & r_size_policy,
    const Trigger_Policy & r_trigger_policy)
```

constructor taking some policies. r\_size\_policy will be copied by the Size\_Policy object of this object. r\_trigger\_policy will be copied by the Trigger\_Policy object of this object.

### 5.518.4 Member Function Documentation

#### get\_actual\_size()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >↵
::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size↵
Access, Size_Type >::get_actual_size () const [inline]
```

Returns the actual size of the container.

#### get\_loads()

```
template<bool External_Load_Access, typename Size_Type>
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size↵
_Type >::get_loads () const [inline], [inherited]
```

Returns a pair of the minimal and maximal loads, respectively.

#### get\_new\_size()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >↵
::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size↵
Access, Size_Type >::get_new_size (
```

```
size_type size,
size_type num_used_e) const [protected]
```

Queries what the new size should be, when the container is resized naturally. The current `__size` of the container is `size`, and the number of used entries within the container is `num_used_e`.

### **get\_size\_policy()** [1/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
Size_Access, Size_Type >::get_size_policy ()
```

Access to the `Size_Policy` object used.

### **get\_size\_policy()** [2/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
const Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
_Size_Access, Size_Type >::get_size_policy () const
```

Const access to the `Size_Policy` object used.

### **get\_trigger\_policy()** [1/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
Size_Access, Size_Type >::get_trigger_policy ()
```

Access to the `Trigger_Policy` object used.

### **get\_trigger\_policy()** [2/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy,
External_Size_Access, Size_Type >::get_trigger_policy () const
```

Access to the `Trigger_Policy` object used.

### **resize()**

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,
Size_Type >::resize (
```

```
size_type suggested_new_size)
```

Resizes the container to `suggested_new_size`, a suggested size (the actual size will be determined by the `Size_Policy` object).

### **set\_loads()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
std::pair< float, float > load_pair) [inherited]
```

Sets the loads through a pair of the minimal and maximal loads, respectively.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)



## 5.519 `std::chrono::hh_mm_ss<_Duration>` Class Template Reference

```
#include <chrono>
```

### Public Types

- using **precision**

### Public Member Functions

- constexpr **hh\_mm\_ss** (`_Duration __d`)
- constexpr `chrono::hours` **hours** () const noexcept
- constexpr bool **is\_negative** () const noexcept
- constexpr `chrono::minutes` **minutes** () const noexcept
- constexpr **operator precision** () const noexcept
- constexpr `chrono::seconds` **seconds** () const noexcept
- constexpr `precision` **subseconds** () const noexcept
- constexpr `precision` **to\_duration** () const noexcept

### Static Public Attributes

- static constexpr unsigned **fractional\_width**

#### 5.519.1 Detailed Description

```
template<typename _Duration>  
class std::chrono::hh_mm_ss<_Duration>
```

Utility for splitting a duration into hours, minutes, and seconds

This is a convenience type that provides accessors for the constituent parts (hours, minutes, seconds and subseconds) of a duration.

Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

## 5.520 `std::locale::id` Class Reference

```
#include <locale_classes.h>
```

### Public Member Functions

- `id` ()
- `size_t _M_id` () const throw ()

### Friends

- template<typename \_Facet>  
const \_Facet \* **\_\_try\_use\_facet** (const `locale` &\_\_loc)
- template<typename \_Facet>  
bool **has\_facet** (const `locale` &\_\_loc) throw ()
- class **locale**
- class **locale::\_Impl**
- template<typename \_Facet>  
const \_Facet & **use\_facet** (const `locale` &\_\_loc)

### 5.520.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

### 5.520.2 Constructor & Destructor Documentation

#### id()

```
std::locale::id::id () [inline]
```

Constructor.

### 5.520.3 Friends And Related Symbol Documentation

#### has\_facet

```
template<typename _Facet>
bool has_facet (
    const locale & __loc) throw ( ) [friend]
```

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

#### Template Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>_Facet</code> | The facet type to test the presence of. |
|---------------------|-----------------------------------------|

#### Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__loc</code> | The locale to test. |
|--------------------|---------------------|

#### Returns

true if `__loc` contains a facet of type `_Facet`, else false.

#### use\_facet

```
template<typename _Facet>
const _Facet & use_facet (
    const locale & __loc) [friend]
```

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

#### Template Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>_Facet</code> | The facet type to access. |
|---------------------|---------------------------|

#### Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__loc</code> | The locale to use. |
|--------------------|--------------------|

#### Returns

Reference to facet of type `Facet`.

#### Exceptions

|                            |                                                                             |
|----------------------------|-----------------------------------------------------------------------------|
| <code>std::bad_cast</code> | if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> . |
|----------------------------|-----------------------------------------------------------------------------|

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 5.521 std::thread::id Class Reference

```
#include <thread>
```

### Public Member Functions

- `id` (`native_handle_type __id`)

### Friends

- struct `hash< id >`
- template<class `_CharT`, class `_Traits`>  
[basic\\_ostream](#)< `_CharT`, `_Traits` > & `operator`<< ([basic\\_ostream](#)< `_CharT`, `_Traits` > &\_\_out, `id __id`)
- strong\_ordering `operator`<=> (`id __x`, `id __y`) noexcept
- bool `operator`== (`id __x`, `id __y`) noexcept
- class `thread`

### Related Symbols

(Note that these are not member symbols.)

- bool `operator`== ([thread::id \\_\\_x](#), [thread::id \\_\\_y](#)) noexcept
- strong\_ordering `operator`<=> ([thread::id \\_\\_x](#), [thread::id \\_\\_y](#)) noexcept
- template<class `_CharT`, class `_Traits`>  
[basic\\_ostream](#)< `_CharT`, `_Traits` > & `operator`<< ([basic\\_ostream](#)< `_CharT`, `_Traits` > &\_\_out, [thread::id \\_\\_id](#))

### 5.521.1 Detailed Description

A `std::thread::id` is a unique identifier for a thread.

Since

C++11

The documentation for this class was generated from the following files:

- [std\\_thread.h](#)
- [thread](#)

## 5.522 std::identity Struct Reference

```
#include <ranges_cmp.h>
```

### Public Types

- using `is_transparent`

### Public Member Functions

- `template<typename _Tp>`  
`constexpr _Tp && operator() (_Tp &&__t) const noexcept`

#### 5.522.1 Detailed Description

[func.identity] The identity function.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 5.523 std::experimental::fundamentals\_v1::in\_place\_t Struct Reference

```
#include <optional>
```

#### 5.523.1 Detailed Description

Tag type for in-place construction.

The documentation for this struct was generated from the following file:

- [experimental/optional](#)

## 5.524 std::independent\_bits\_engine<\_RandomNumberEngine, \_\_w, \_UIntType > Class Template Reference

```
#include <random>
```

### Public Types

- typedef \_UIntType `result_type`

### Public Member Functions

- `independent_bits_engine ()`
- `independent_bits_engine (_RandomNumberEngine &&__rng)`
- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`  
`independent_bits_engine (_Sseq &__q)`
- `independent_bits_engine (const _RandomNumberEngine &__rng)`
- `independent_bits_engine (result_type __s)`
- `const _RandomNumberEngine & base () const noexcept`
- `void discard (unsigned long long __z)`
- `result_type operator() ()`
- `void seed ()`
- `template<typename _Sseq>`  
`_If_seed_seq<_Sseq> > seed (_Sseq &__q)`
- `void seed (result_type __s)`

## Static Public Member Functions

- static constexpr [result\\_type](#) max ()
- static constexpr [result\\_type](#) min ()

## Friends

- bool [operator==](#) (const [independent\\_bits\\_engine](#) &\_\_lhs, const [independent\\_bits\\_engine](#) &\_\_rhs)
- template<typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is,  
[std::independent\\_bits\\_engine](#)< \_RandomNumberEngine, \_\_w, \_UIntType > &\_\_x)

### 5.524.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specified number of bits \_\_w.

Since

C++11

### 5.524.2 Member Typedef Documentation

#### result\_type

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
typedef _UIntType std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::result_type
```

The type of the generated random value.

### 5.524.3 Constructor & Destructor Documentation

#### independent\_bits\_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ()
[inline]
```

Constructs a default [independent\\_bits\\_engine](#) engine.

The underlying engine is default constructed as well.

Referenced by [operator==](#).

#### independent\_bits\_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    const _RandomNumberEngine & __rng) [inline], [explicit]
```

Copy constructs a [independent\\_bits\\_engine](#) engine.

Copies an existing base class random number generator.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

**independent\_bits\_engine()** [3/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    _RandomNumberEngine && __rng) [inline], [explicit]
```

Move constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

References [std::move\(\)](#).

**independent\_bits\_engine()** [4/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    result_type __s) [inline], [explicit]
```

Seed constructs a `independent_bits_engine` engine.

Constructs the underlying generator engine seeded with `__s`.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A seed value for the base class engine. |
|------------------|-----------------------------------------|

**independent\_bits\_engine()** [5/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    _Sseq & __q) [inline], [explicit]
```

Generator construct a `independent_bits_engine` engine.

**Parameters**

|                  |                  |
|------------------|------------------|
| <code>__q</code> | A seed sequence. |
|------------------|------------------|

**5.524.4 Member Function Documentation****base()**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
const _RandomNumberEngine & std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::
base () const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

**discard()**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::discard (
    unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

**max()**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
static constexpr result\_type std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType
>::max () [inline], [static], [constexpr]
```

Gets the maximum value in the generated random number range.

**min()**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
static constexpr result\_type std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType
>::min () [inline], [static], [constexpr]
```

Gets the minimum value in the generated random number range.

**operator()()**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::result_type std::independent\_bits\_engine<
_RandomNumberEngine, __w, _UIntType >::operator() ()
```

Gets the next value in the generated random number sequence.

References [std::\\_\\_lg\(\)](#), [std::\\_\\_numeric\\_limits\\_base::digits](#), and [std::numeric\\_limits<\\_Tp>::max\(\)](#).

**seed()** [1/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::seed () [inline]
```

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

**seed()** [2/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::seed
(
    _Sseq & __q) [inline]
```

Reseeds the `independent_bits_engine` object with the given seed sequence.

**Parameters**

|                  |                            |
|------------------|----------------------------|
| <code>__q</code> | A seed generator function. |
|------------------|----------------------------|

**seed()** [3/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::seed (
    result\_type __s) [inline]
```

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

**5.524.5 Friends And Related Symbol Documentation****operator==**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
bool operator== (
```

```
const independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __lhs,
const independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __rhs) [friend]
```

Compares two independent\_bits\_engine random number generator objects of the same type for equality.

#### Parameters

|                    |                                                                 |
|--------------------|-----------------------------------------------------------------|
| <code>__lhs</code> | A independent_bits_engine random number generator object.       |
| <code>__rhs</code> | Another independent_bits_engine random number generator object. |

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

References [independent\\_bits\\_engine\(\)](#).

#### operator>>

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (
    std::basic_istream<_CharT, _Traits> & __is,
    std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __x) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

#### Parameters

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| <code>__is</code> | An input stream.                                          |
| <code>__x</code>  | A independent_bits_engine random number generator engine. |

#### Returns

The input stream with the state of \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.525 std::indirect\_array<\_Tp> Class Template Reference

```
#include <indirect_array.h>
```

#### Public Types

- typedef `_Tp` **value\_type**

#### Public Member Functions

- [indirect\\_array](#) (const [indirect\\_array](#) &)
- template<class `_Dom`>  
void **operator%=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void [operator%=](#) (const [valarray](#)< `_Tp` > &) const



- `template<class _Dom>`  
`void operator&= (const _Expr< _Dom, _Tp > &) const`
- `void operator&= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator* = (const _Expr< _Dom, _Tp > &) const`
- `void operator* = (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator+= (const _Expr< _Dom, _Tp > &) const`
- `void operator+= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator-= (const _Expr< _Dom, _Tp > &) const`
- `void operator-= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator/= (const _Expr< _Dom, _Tp > &) const`
- `void operator/= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator<<= (const _Expr< _Dom, _Tp > &) const`
- `void operator<<= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator= (const _Expr< _Dom, _Tp > &) const`
- `void operator= (const _Tp &) const`
- `indirect_array & operator= (const indirect_array &)`
- `void operator= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator|= (const _Expr< _Dom, _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`

## Friends

- `class gslice_array< _Tp >`
- `class valarray< _Tp >`

### 5.525.1 Detailed Description

`template<class _Tp>`

`class std::indirect_array< _Tp >`

Reference to arbitrary subset of an array.

An `indirect_array` is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an `indirect_array` is to call `operator[]`(`valarray<size_t>`) on a `valarray`. The returned `indirect_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if an `indirect_array` is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

#### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

### 5.525.2 Member Function Documentation

#### operator%=( )

```
template<class _Tp>
void std::indirect_array< _Tp >::operator%= (
    const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

#### operator&=( )

```
template<class _Tp>
void std::indirect_array< _Tp >::operator&= (
    const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

#### operator\*=( )

```
template<class _Tp>
void std::indirect_array< _Tp >::operator*= (
    const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

#### operator+=( )

```
template<class _Tp>
void std::indirect_array< _Tp >::operator+= (
    const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

#### operator-=( )

```
template<class _Tp>
void std::indirect_array< _Tp >::operator-= (
    const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

#### operator/=( )

```
template<class _Tp>
void std::indirect_array< _Tp >::operator/= (
    const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

#### operator<<=( )

```
template<class _Tp>
void std::indirect_array< _Tp >::operator<<= (
    const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

#### operator>>=( )

```
template<class _Tp>
void std::indirect_array< _Tp >::operator>>= (
    const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.  
References [indirect\\_array\(\)](#).

### **operator^=()**

```
template<class _Tp>
void std::indirect_array< _Tp >::operator^= (
    const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

### **operator" |=()**

```
template<class _Tp>
void std::indirect_array< _Tp >::operator|= (
    const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following files:

- [valarray](#)
- [indirect\\_array.h](#)

## **5.526 std::initializer\_list< \_E > Class Template Reference**

```
#include <initializer_list>
```

### **Public Types**

- typedef const \_E \* **const\_iterator**
- typedef const \_E & **const\_reference**
- typedef const \_E \* **iterator**
- typedef const \_E & **reference**
- typedef size\_t **size\_type**
- typedef \_E **value\_type**

### **Public Member Functions**

- constexpr const\_iterator **begin** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **size** () const noexcept

### **Related Symbols**

(Note that these are not member symbols.)

- template<class \_Tp>  
constexpr const \_Tp \* **begin** (initializer\_list< \_Tp > \_\_ils) noexcept
- template<class \_Tp>  
constexpr const \_Tp \* **end** (initializer\_list< \_Tp > \_\_ils) noexcept

#### **5.526.1 Detailed Description**

```
template<class _E>
```

```
class std::initializer_list< _E >
```

```
initializer_list
```

### 5.526.2 Friends And Related Symbol Documentation

#### begin()

```
template<class _Tp>
const _Tp * begin (
    initializer_list< _Tp > __ils) [related]
```

Return an iterator pointing to the first element of the initializer\_list.

#### Parameters

|                            |                   |
|----------------------------|-------------------|
| $\leftrightarrow$<br>__ils | Initializer list. |
|----------------------------|-------------------|

#### end()

```
template<class _Tp>
const _Tp * end (
    initializer_list< _Tp > __ils) [related]
```

Return an iterator pointing to one past the last element of the initializer\_list.

#### Parameters

|                            |                   |
|----------------------------|-------------------|
| $\leftrightarrow$<br>__ils | Initializer list. |
|----------------------------|-------------------|

The documentation for this class was generated from the following file:

- [initializer\\_list](#)

### 5.527 std::input\_iterator\_tag Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for `std::input_iterator_tag`:



#### 5.527.1 Detailed Description

Marking input iterators.

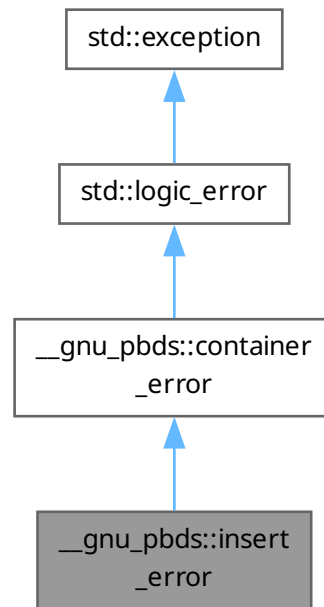
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 5.528 `__gnu_pbds::insert_error` Struct Reference

```
#include <exception.hpp>
```

Inheritance diagram for `__gnu_pbds::insert_error`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 5.528.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the `allocator_type`'s exception will be thrown).

#### 5.528.2 Member Function Documentation

##### `what()`

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

### 5.529 `std::insert_iterator<_Container>` Class Template Reference

```
#include <stl_iterator.h>
```

Inheritance diagram for `std::insert_iterator< _Container >`:



### Public Types

- typedef `_Container` `container_type`
- using `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

### Public Member Functions

- constexpr `insert_iterator` (`_Container &__x`, `_Iter __i`)
- constexpr `insert_iterator` & `operator*` ()
- constexpr `insert_iterator` & `operator++` ()
- constexpr `insert_iterator` & `operator++` (int)
- constexpr `insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- constexpr `insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_value)

### Protected Attributes

- `_Container *` `container`
- `_Iter` `iter`

#### 5.529.1 Detailed Description

**template<typename \_Container>**  
**class** `std::insert_iterator< _Container >`

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

### 5.529.2 Member Typedef Documentation

#### container\_type

```
template<typename _Container>
typedef _Container std::insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

#### iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator←_category [inherited]
```

One of the [tag types](#).

#### pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

#### reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value\_type.

#### value\_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
```

The type "pointed to" by the iterator.

### 5.529.3 Constructor & Destructor Documentation

#### insert\_iterator()

```
template<typename _Container>
std::insert_iterator< _Container >::insert_iterator (
    _Container & __x,
    _Iter __i) [inline], [constexpr]
```

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

References [std::\\_\\_addressof\(\)](#).

### 5.529.4 Member Function Documentation

#### operator\*()

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator* () [inline], [nodiscard], [constexpr]
```

Simply returns \*this.

#### operator++() [1/2]

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator++ () [inline], [constexpr]
```

Simply returns \*this. (This iterator does not *move*.)



**operator++()** [2/2]

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator++ (
    int ) [inline], [constexpr]
```

Simply returns \*this. (This iterator does not *move*.)

**operator=()**

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator= (
    const typename _Container::value_type & __value) [inline], [constexpr]
```

**Parameters**

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z
insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;
```

```
// vector v contains A, 1, 2, 3, and Z
```

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

**5.530 std::integer\_sequence< \_Tp, \_Idx > Struct Template Reference**

```
#include <utility.h>
```

**Public Types**

- `typedef _Tp value_type`

**Static Public Member Functions**

- `static constexpr size_t size () noexcept`

**5.530.1 Detailed Description**

```
template<typename _Tp, _Tp... _Idx>
struct std::integer_sequence< _Tp, _Idx >
```

Class template `integer_sequence`.

The documentation for this struct was generated from the following file:

- [utility.h](#)

## 5.531 std::integral\_constant< \_Tp, \_\_v > Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.531.1 Detailed Description

```
template<typename _Tp, _Tp __v>  
struct std::integral_constant< _Tp, __v >
```

integral\_constant

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.532 std::invalid\_argument Class Reference

```
#include <stdexcept>
```

Inheritance diagram for std::invalid\_argument:



### Public Member Functions

- **invalid\_argument** (const char \*)
- **invalid\_argument** (const [invalid\\_argument](#) &)=default

- `invalid_argument` (const [string](#) &\_\_arg)
- `invalid_argument` ([invalid\\_argument](#) &&)=default
- `invalid_argument` & `operator=` (const [invalid\\_argument](#) &)=default
- `invalid_argument` & `operator=` ([invalid\\_argument](#) &&)=default
- virtual const char \* [what](#) () const noexcept

#### 5.532.1 Detailed Description

Thrown to report invalid arguments to functions.

#### 5.532.2 Member Function Documentation

##### `what()`

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 5.533 `std::ios_base` Class Reference

```
#include <ios_base.h>
```

[illegible]

- enum [event](#) { [erase\\_event](#) , [imbue\\_event](#) , [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
- typedef \_ios\_losestate [iostate](#)
- typedef \_ios\_Openmode [openmode](#)
- typedef \_ios\_Seekdir [seekdir](#)

- **ios\_base** (const **ios\_base** &)=delete
- virtual ~**ios\_base** ()
- const **locale** & **\_M\_getloc** () const
- **fmtflags** **flags** () const

- `fmtflags flags (fmtflags __fmtfl)`
- `locale getloc () const`
- `locale imbue (const locale &__loc) throw ()`
- `long & iword (int __ix)`
- `ios_base & operator= (const ios_base &)=delete`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `void register_callback (event_callback __fn, int __index)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void unsetf (fmtflags __mask)`
- `streamsize width () const`
- `streamsize width (streamsize __wide)`

### Static Public Member Functions

- `static bool sync_with_stdio (bool __sync=true)`
- `static int xalloc () throw ()`

### Public Attributes

- `class __attribute__((__abi_tag__("cxx11"))) failure typedef _los_Fmtflags fmtflags`

### Static Public Attributes

- `static const openmode __noreplace`
- `static const fmtflags adjustfield`
- `static const openmode app`
- `static const openmode ate`
- `static const iostate badbit`
- `static const fmtflags basefield`
- `static const seekdir beg`
- `static const openmode binary`
- `static const fmtflags boolalpha`
- `static const seekdir cur`
- `static const fmtflags dec`
- `static const seekdir end`
- `static const iostate eofbit`
- `static const iostate failbit`
- `static const fmtflags fixed`
- `static const fmtflags floatfield`
- `static const iostate goodbit`
- `static const fmtflags hex`
- `static const openmode in`
- `static const fmtflags internal`
- `static const fmtflags left`
- `static const fmtflags oct`
- `static const openmode out`
- `static const fmtflags right`
- `static const fmtflags scientific`
- `static const fmtflags showbase`

- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) &\_\_rhs) noexcept

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- [iostate](#) [\\_M\\_exception](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- [streamsize](#) [\\_M\\_precision](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

#### 5.533.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

#### 5.533.2 Member Typedef Documentation

##### event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios\_base &__b, int __i)
```

The type of an event callback function.

##### Parameters

|                     |                                                |
|---------------------|------------------------------------------------|
| <a href="#">__e</a> | One of the members of the event enum.          |
| <a href="#">__b</a> | Reference to the <code>ios_base</code> object. |

|                 |                                                        |
|-----------------|--------------------------------------------------------|
| <code>_i</code> | The integer provided when the callback was registered. |
|-----------------|--------------------------------------------------------|

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

### **iostate**

```
typedef _Ios_Iostate std::ios_base::iostate
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

### **openmode**

```
typedef _Ios_Openmode std::ios_base::openmode
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

### **seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

## **5.533.3 Member Enumeration Documentation**

### **event**

```
enum std::ios_base::event
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

#### 5.533.4 Constructor & Destructor Documentation

##### ~ios\_base()

```
virtual std::ios_base::~~ios_base () [virtual]
```

Invokes each callback with `erase_event`. Destroys local storage.

Note that the `ios_base` object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with `erase_event` (unless `copyfmt` is used).

#### 5.533.5 Member Function Documentation

##### \_M\_getloc()

```
const locale & std::ios_base::_M_getloc () const [inline]
```

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_get< _CharT, _Inlter >::do_get_year()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

##### flags() [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard]
```

Access to format flags.

##### Returns

The format control flags for both input and output.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::subtract_with_carry_engine< uint_fast32_t >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::subtract_with_carry_engine< uint_fast32_t >::operator>>()`.

##### flags() [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline]
```

Setting new format flags all at once.

##### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

##### Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

References [fmtflags](#).



**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::money\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::do\\_get\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

**imbue()**

```
locale std::ios_base::imbue (
    const locale & __loc) throw ( )
```

Setting a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Sets the new locale for this stream, and then invokes each callback with `imbue_event`.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::imbue\(\)](#).

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [iword\(\)](#).

Referenced by [iword\(\)](#).

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), and [std::chrono::operator<<\(\)](#).

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index)
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <code>__fn</code> | The function to add. |
|-------------------|----------------------|

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__index</code> | The integer to pass to the function when invoked. |
|----------------------|---------------------------------------------------|

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered. References [fmtflags](#).

#### **setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline]
```

Setting new format flags.

##### Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

##### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

References [fmtflags](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

#### **setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline]
```

Setting new format flags.

##### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

##### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

References [fmtflags](#).

#### **sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static]
```

Interaction with the standard C I/O objects.

##### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

References [fmtflags](#).

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::num\\_put<\\_CharT, \\_Outiter>::do\\_put\(\)](#), [std::operator>>\(\)](#), and [std::operator>>\(\)](#).

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static]
```

Access to unique indices.

## Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.533.6 Member Data Documentation

### **adjustfield**

```
const fmtflags std::ios_base::adjustfield [static]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::internal\(\)](#), [std::left\(\)](#), and [std::right\(\)](#).

### **app**

```
const openmode std::ios_base::app [static]
```

Seek to end before each write.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

### **ate**

```
const openmode std::ios_base::ate [static]
```

Open and seek to end immediately after opening.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::open\(\)](#).

### **badbit**

```
const iostate std::ios_base::badbit [static]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::~sentry\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::clear\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::read\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::tellg\(\)](#), and [std::ws\(\)](#).

### **basefield**

```
const fmtflags std::ios_base::basefield [static]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Referenced by [std::dec\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::hex\(\)](#), and [std::oct\(\)](#).

### **beg**

```
const seekdir std::ios_base::beg [static]
```

Request a seek relative to the beginning of the stream.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekpos\(\)](#).

**binary**

```
const openmode std::ios_base::binary [static]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary>.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::showmanyc\(\)](#).

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

**cur**

```
const seekdir std::ios_base::cur [static]
```

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::imbue\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), and [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#).

**dec**

```
const fmtflags std::ios_base::dec [static]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#).

**end**

```
const seekdir std::ios_base::end [static]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::open\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_date\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_time\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::g](#), [std::time\\_get<\\_CharT, \\_InIter>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::read\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::read\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<char\\_type, traits\\_type>::seekg\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), and [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_weekday\(\)](#).

`std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< char_type, traits_type >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::seekp()`.

## fixed

const `fmtflags` `std::ios_base::fixed` [static]

Generate floating-point output in fixed-point notation.

Referenced by `std::fixed()`, and `std::hexfloat()`.

## floatfield

const `fmtflags` `std::ios_base::floatfield` [static]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

## goodbit

const `iosstate` `std::ios_base::goodbit` [static]

Indicates all is well.

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::ws()`.

## hex

const `fmtflags` `std::ios_base::hex` [static]

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::hex()`.

## in

const `openmode` `std::ios_base::in` [static]

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::xsgetn()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

## internal

const `fmtflags` `std::ios_base::internal` [static]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

### **left**

```
const fmtflags std::ios_base::left [static]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by [std::num\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), and [std::left\(\)](#).

### **oct**

```
const fmtflags std::ios_base::oct [static]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

### **out**

```
const openmode std::ios_base::out [static]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

### **right**

```
const fmtflags std::ios_base::right [static]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Referenced by [std::right\(\)](#).

### **scientific**

```
const fmtflags std::ios_base::scientific [static]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

### **showbase**

```
const fmtflags std::ios_base::showbase [static]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

### **showpoint**

```
const fmtflags std::ios_base::showpoint [static]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

### **showpos**

```
const fmtflags std::ios_base::showpos [static]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).



**skipws**

```
const fmtflags std::ios_base::skipws [static]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), and [std::skipws\(\)](#).

**trunc**

```
const openmode std::ios_base::trunc [static]
```

Truncate an existing stream when opening. Default for `ofstream`.

**unitbuf**

```
const fmtflags std::ios_base::unitbuf [static]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

```
const fmtflags std::ios_base::uppercase [static]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

**5.534 std::is\_abstract<\_Tp> Struct Template Reference**

```
#include <type_traits>
```

**5.534.1 Detailed Description**

```
template<typename _Tp>
```

```
struct std::is_abstract<_Tp>
```

`is_abstract`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.535 std::is\_arithmetic<\_Tp> Struct Template Reference**

```
#include <type_traits>
```

**5.535.1 Detailed Description**

```
template<typename _Tp>
```

```
struct std::is_arithmetic<_Tp>
```

`is_arithmetic`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.536 std::is\_array<\_Tp> Struct Template Reference**

```
#include <type_traits>
```

#### 5.536.1 Detailed Description

```
template<typename _Tp>
struct std::is_array< _Tp >
```

`is_array`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.537 `std::is_assignable< _Tp, _Up >` Struct Template Reference

```
#include <type_traits>
```

#### 5.537.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_assignable< _Tp, _Up >
```

`is_assignable`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.538 `std::is_base_of< _Base, _Derived >` Struct Template Reference

```
#include <type_traits>
```

#### 5.538.1 Detailed Description

```
template<typename _Base, typename _Derived>
struct std::is_base_of< _Base, _Derived >
```

`is_base_of`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.539 `std::is_bind_expression< _Tp >` Struct Template Reference

```
#include <functional>
```

#### 5.539.1 Detailed Description

```
template<typename _Tp>
struct std::is_bind_expression< _Tp >
```

Trait that identifies a bind expression.

Determines if the given type `_Tp` is a function object that should be treated as a subexpression when evaluating calls to function objects returned by `std::bind`.

C++11 [func.bind.isbind].

Since

C++11

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.540 `std::is_bind_expression< _Bind< _Signature > >` Struct Template Reference

```
#include <functional>
```

#### 5.540.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.541 `std::is_bind_expression< _Bind_result< _Result, _Signature > >` Struct Template Reference

```
#include <functional>
```

#### 5.541.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.542 `std::is_bind_expression< const _Bind< _Signature > >` Struct Template Reference

```
#include <functional>
```

#### 5.542.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< const _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.543 `std::is_bind_expression< const _Bind_result< _Result, _Signature > >` Struct Template Reference

```
#include <functional>
```

#### 5.543.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.544 `std::is_bind_expression< const volatile _Bind< _Signature > >` Struct Template Reference

```
#include <functional>
```

#### 5.544.1 Detailed Description

```
template<typename _Signature>  
struct std::is_bind_expression< const volatile _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.545 `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >` Struct Template Reference

```
#include <functional>
```

#### 5.545.1 Detailed Description

```
template<typename _Result, typename _Signature>  
struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.546 `std::is_bind_expression< volatile _Bind< _Signature > >` Struct Template Reference

```
#include <functional>
```

#### 5.546.1 Detailed Description

```
template<typename _Signature>  
struct std::is_bind_expression< volatile _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.547 `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >` Struct Template Reference

```
#include <functional>
```

#### 5.547.1 Detailed Description

```
template<typename _Result, typename _Signature>  
struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.548 `std::is_class<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.548.1 Detailed Description

```
template<typename _Tp>  
struct std::is_class<_Tp>
```

`is_class`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.549 `std::is_compound<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.549.1 Detailed Description

```
template<typename _Tp>  
struct std::is_compound<_Tp>
```

`is_compound`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.550 `std::is_const<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.550.1 Detailed Description

```
template<typename _Tp>  
struct std::is_const<_Tp>
```

`is_const`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.551 `std::is_constructible<_Tp, _Args>` Struct Template Reference

```
#include <type_traits>
```

#### 5.551.1 Detailed Description

```
template<typename _Tp, typename... _Args>  
struct std::is_constructible<_Tp, _Args>
```

`is_constructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.552 `std::is_copy_assignable<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.552.1 Detailed Description

```
template<typename _Tp>
struct std::is_copy_assignable<_Tp>
```

`is_copy_assignable`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.553 `std::is_copy_constructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.553.1 Detailed Description

```
template<typename _Tp>
struct std::is_copy_constructible<_Tp>
```

`is_copy_constructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.554 `std::is_default_constructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.554.1 Detailed Description

```
template<typename _Tp>
struct std::is_default_constructible<_Tp>
```

`is_default_constructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.555 `std::is_destructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.555.1 Detailed Description

```
template<typename _Tp>
struct std::is_destructible<_Tp>
```

`is_destructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.556 `std::is_empty<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.556.1 Detailed Description

```
template<typename _Tp>
struct std::is_empty< _Tp >
```

is\_empty

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.557 std::is\_enum< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### 5.557.1 Detailed Description

```
template<typename _Tp>
struct std::is_enum< _Tp >
```

is\_enum

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.558 std::is\_error\_code\_enum< \_Tp > Struct Template Reference

```
#include <system_error>
```

##### 5.558.1 Detailed Description

```
template<typename _Tp>
struct std::is_error_code_enum< _Tp >
```

is\_error\_code\_enum

The documentation for this struct was generated from the following file:

- [system\\_error](#)

#### 5.559 std::is\_error\_code\_enum< future\_errc > Struct Reference

```
#include <future>
```

##### 5.559.1 Detailed Description

Specialization that allows `future_errc` to convert to `error_code`.

The documentation for this struct was generated from the following file:

- [future](#)

#### 5.560 std::is\_error\_condition\_enum< \_Tp > Struct Template Reference

```
#include <system_error>
```

### 5.560.1 Detailed Description

```
template<typename _Tp>
struct std::is_error_condition_enum< _Tp >
```

is\_error\_condition\_enum

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.561 std::is\_floating\_point< \_Tp > Struct Template Reference

```
#include <type_traits>
```

Inheritance diagram for std::is\_floating\_point< \_Tp >:



### 5.561.1 Detailed Description

```
template<typename _Tp>
struct std::is_floating_point< _Tp >
```

is\_floating\_point

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.562 std::is\_function< \_Tp > Struct Template Reference

```
#include <type_traits>
```

### 5.562.1 Detailed Description

```
template<typename _Tp>
struct std::is_function< _Tp >
```

is\_function

The documentation for this struct was generated from the following file:

- [type\\_traits](#)



### 5.563 `std::is_fundamental<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.563.1 Detailed Description

```
template<typename _Tp>
struct std::is_fundamental<_Tp>
```

`is_fundamental`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.564 `std::is_integral<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.564.1 Detailed Description

```
template<typename _Tp>
struct std::is_integral<_Tp>
```

`is_integral`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.565 `std::is_layout_compatible<_Tp, _Up>` Struct Template Reference

```
#include <type_traits>
```

#### 5.565.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_layout_compatible<_Tp, _Up>
```

- Remove references and cv-qualifiers.

Since

C++20

C++20

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.566 `std::is_literal_type<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.566.1 Detailed Description

```
template<typename _Tp>
struct std::is_literal_type<_Tp>
```

`is_literal_type`

**Deprecated** Deprecated in C++17, removed in C++20. The idea of a literal type isn't useful.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.567 `std::is_lvalue_reference< typename >` Struct Template Reference

```
#include <type_traits>
```

### 5.567.1 Detailed Description

```
template<typename>  
struct std::is_lvalue_reference< typename >
```

`is_lvalue_reference`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.568 `std::is_member_function_pointer< _Tp >` Struct Template Reference

```
#include <type_traits>
```

### 5.568.1 Detailed Description

```
template<typename _Tp>  
struct std::is_member_function_pointer< _Tp >
```

`is_member_function_pointer`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.569 `std::is_member_object_pointer< _Tp >` Struct Template Reference

```
#include <type_traits>
```

### 5.569.1 Detailed Description

```
template<typename _Tp>  
struct std::is_member_object_pointer< _Tp >
```

`is_member_object_pointer`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.570 `std::is_member_pointer< _Tp >` Struct Template Reference

```
#include <type_traits>
```

### 5.570.1 Detailed Description

```
template<typename _Tp>  
struct std::is_member_pointer< _Tp >
```

`is_member_pointer`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.571 std::is\_move\_assignable< \_Tp > Struct Template Reference**

```
#include <type_traits>
```

**5.571.1 Detailed Description**

```
template<typename _Tp>
struct std::is_move_assignable< _Tp >
```

is\_move\_assignable

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.572 std::is\_move\_constructible< \_Tp > Struct Template Reference**

```
#include <type_traits>
```

**5.572.1 Detailed Description**

```
template<typename _Tp>
struct std::is_move_constructible< _Tp >
```

is\_move\_constructible

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.573 std::is\_nothrow\_assignable< \_Tp, \_Up > Struct Template Reference**

```
#include <type_traits>
```

**5.573.1 Detailed Description**

```
template<typename _Tp, typename _Up>
struct std::is_nothrow_assignable< _Tp, _Up >
```

is\_nothrow\_assignable

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.574 std::is\_nothrow\_constructible< \_Tp, \_Args > Struct Template Reference**

```
#include <type_traits>
```

**5.574.1 Detailed Description**

```
template<typename _Tp, typename... _Args>
struct std::is_nothrow_constructible< _Tp, _Args >
```

is\_nothrow\_constructible

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.575 std::is\_nothrow\_copy\_assignable< \_Tp > Struct Template Reference**

```
#include <type_traits>
```

#### 5.575.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_copy_assignable<_Tp>
```

`is_nothrow_copy_assignable`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.576 `std::is_nothrow_copy_constructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.576.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_copy_constructible<_Tp>
```

`is_nothrow_copy_constructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.577 `std::is_nothrow_default_constructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.577.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_default_constructible<_Tp>
```

`is_nothrow_default_constructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.578 `std::is_nothrow_destructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.578.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_destructible<_Tp>
```

`is_nothrow_destructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.579 `std::is_nothrow_move_assignable<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### 5.579.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_move_assignable< _Tp >
```

is\_nothrow\_move\_assignable

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.580 std::is\_nothrow\_move\_constructible< \_Tp > Struct Template Reference

```
#include <type_traits>
```

### 5.580.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_move_constructible< _Tp >
```

is\_nothrow\_move\_constructible

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.581 std::is\_object< \_Tp > Struct Template Reference

```
#include <type_traits>
```

### 5.581.1 Detailed Description

```
template<typename _Tp>
struct std::is_object< _Tp >
```

is\_object

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.582 std::is\_placeholder< \_Tp > Struct Template Reference

```
#include <functional>
```

Inheritance diagram for `std::is_placeholder<_Tp>`:



### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr int **value**

#### 5.582.1 Detailed Description

```
template<typename _Tp>
struct std::is_placeholder<_Tp>
```

Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is.  
C++11 [func.bind.isplace].

Since

C++11

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.583 `std::is_placeholder<_Placeholder<_Num>>` Struct Template Reference

```
#include <functional>
```

Inheritance diagram for `std::is_placeholder<_Placeholder<_Num>>`:



### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr int **value**

#### 5.583.1 Detailed Description

```
template<int _Num>
struct std::is_placeholder<_Placeholder<_Num>>
```

Partial specialization of `is_placeholder` that provides the placeholder number for the placeholder objects defined by `libstdc++`.

Since

C++11

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.584 std::is\_pod<\_Tp> Struct Template Reference

```
#include <type_traits>
```

#### 5.584.1 Detailed Description

```
template<typename _Tp>
struct std::is_pod<_Tp>
```

`is_pod`

**Deprecated** Deprecated in C++20. Use `is_standard_layout` && `is_trivial` instead.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.585 `std::is_pointer< _Tp >` Struct Template Reference

```
#include <type_traits>
```

### 5.585.1 Detailed Description

```
template<typename _Tp>
struct std::is_pointer< _Tp >
```

`is_pointer`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.586 `std::is_pointer_interconvertible_base_of< _Base, _Derived >` Struct Template Reference

```
#include <type_traits>
```

### 5.586.1 Detailed Description

```
template<typename _Base, typename _Derived>
struct std::is_pointer_interconvertible_base_of< _Base, _Derived >
```

True if `_Derived` is standard-layout and has a base class of type `_Base`

Since

C++20

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.587 `std::is_polymorphic< _Tp >` Struct Template Reference

```
#include <type_traits>
```

### 5.587.1 Detailed Description

```
template<typename _Tp>
struct std::is_polymorphic< _Tp >
```

`is_polymorphic`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.588 `std::is_reference< _Tp >` Struct Template Reference

```
#include <type_traits>
```



#### 5.588.1 Detailed Description

```
template<typename _Tp>
struct std::is_reference< _Tp >
```

is\_reference

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.589 std::is\_rvalue\_reference< typename > Struct Template Reference

```
#include <type_traits>
```

#### 5.589.1 Detailed Description

```
template<typename>
struct std::is_rvalue_reference< typename >
```

is\_rvalue\_reference

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.590 std::is\_same< \_Tp, \_Up > Struct Template Reference

```
#include <type_traits>
```

#### 5.590.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_same< _Tp, _Up >
```

is\_same

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.591 std::is\_scalar< \_Tp > Struct Template Reference

```
#include <type_traits>
```

#### 5.591.1 Detailed Description

```
template<typename _Tp>
struct std::is_scalar< _Tp >
```

is\_scalar

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.592 std::is\_signed< \_Tp > Struct Template Reference

```
#include <type_traits>
```

#### 5.592.1 Detailed Description

```
template<typename _Tp>
struct std::is_signed< _Tp >
```

is\_signed

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.593 std::is\_standard\_layout< \_Tp > Struct Template Reference

```
#include <type_traits>
```

#### 5.593.1 Detailed Description

```
template<typename _Tp>
struct std::is_standard_layout< _Tp >
```

is\_standard\_layout

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.594 std::is\_trivial< \_Tp > Struct Template Reference

```
#include <type_traits>
```

#### 5.594.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivial< _Tp >
```

is\_trivial

**Deprecated** Deprecated in C++26. Use a combination of one or more more specialized type traits instead, such as `is_trivially_default_constructible`, `is_trivially_copy_constructible`, `is_trivially_copy_assignable`, etc., depending on the exact check(s) needed.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.595 std::is\_trivially\_assignable< \_Tp, \_Up > Struct Template Reference

```
#include <type_traits>
```

#### 5.595.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_trivially_assignable< _Tp, _Up >
```

is\_trivially\_assignable

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.596 std::is\_trivially\_constructible< \_Tp, \_Args > Struct Template Reference

```
#include <type_traits>
```

#### 5.596.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_trivially_constructible< _Tp, _Args >
```

is\_trivially\_constructible

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.597 std::is\_trivially\_copy\_assignable< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### 5.597.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_copy_assignable< _Tp >
```

is\_trivially\_copy\_assignable

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.598 std::is\_trivially\_copy\_constructible< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### 5.598.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_copy_constructible< _Tp >
```

is\_trivially\_copy\_constructible

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.599 std::is\_trivially\_copyable< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### 5.599.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_copyable< _Tp >
```

is\_trivially\_copyable

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.600 std::is\_trivially\_default\_constructible< \_Tp > Struct Template Reference

```
#include <type_traits>
```

### 5.600.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_default_constructible<_Tp>
```

`is_trivially_default_constructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.601 `std::is_trivially_destructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### 5.601.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_destructible<_Tp>
```

`is_trivially_destructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.602 `std::is_trivially_move_assignable<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### 5.602.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_move_assignable<_Tp>
```

`is_trivially_move_assignable`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.603 `std::is_trivially_move_constructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### 5.603.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_move_constructible<_Tp>
```

`is_trivially_move_constructible`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.604 `std::is_union<_Tp>` Struct Template Reference

```
#include <type_traits>
```

#### 5.604.1 Detailed Description

```
template<typename _Tp>
struct std::is_union< _Tp >
```

is\_union

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.605 std::is\_unsigned< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### 5.605.1 Detailed Description

```
template<typename _Tp>
struct std::is_unsigned< _Tp >
```

is\_unsigned

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.606 std::is\_void< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### 5.606.1 Detailed Description

```
template<typename _Tp>
struct std::is_void< _Tp >
```

is\_void

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.607 std::is\_volatile< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### 5.607.1 Detailed Description

```
template<typename _Tp>
struct std::is_volatile< _Tp >
```

is\_volatile

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.608 std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > Class Template Reference

```
#include <stream_iterator.h>
```

Inheritance diagram for std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `ptrdiff_t` **difference\_type**
- typedef `basic_istream< _CharT, _Traits >` **istream\_type**
- typedef `input_iterator_tag` **iterator\_category**
- typedef `const _Tp *` **pointer**
- typedef `const _Tp &` **reference**
- typedef `_Traits` **traits\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr `istream_iterator` () noexcept(*/\*conditional \*/*)
- constexpr `istream_iterator` (const `istream_iterator` &\_\_obj) noexcept(*/\*conditional \*/*)
- `istream_iterator` (`istream_type` &\_\_s)
- const `_Tp` & **operator\*** () const noexcept
- `istream_iterator` & **operator++** ()
- `istream_iterator` **operator++** (int)
- const `_Tp *` **operator->** () const noexcept
- `istream_iterator` & **operator=** (const `istream_iterator` &)=default

### Friends

- bool **operator==** (const `istream_iterator` &\_\_x, const `istream_iterator` &\_\_y) noexcept

#### 5.608.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist
= ptrdiff_t>
```

```
class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >
```

Provides input iterator semantics for streams.

## 5.608.2 Member Typedef Documentation

### difference\_type

```
typedef ptrdiff_t std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp & >::difference_type [inherited]
```

Distance between iterators is represented as this type.

### iterator\_category

```
typedef input_iterator_tag std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp & >::iterator_category [inherited]
```

One of the [tag types](#).

### pointer

```
typedef const _Tp * std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp & >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

### reference

```
typedef const _Tp & std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp & >::reference [inherited]
```

This type represents a reference-to-value\_type.

### value\_type

```
typedef _Tp std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp & >::value_type [inherited]
```

The type "pointed to" by the iterator.

## 5.608.3 Constructor & Destructor Documentation

### istream\_iterator() [1/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator () [inline], [constexpr], [noexcept]
```

Construct end of input stream iterator.

Referenced by [operator==](#).

### istream\_iterator() [2/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (
    istream_type & __s) [inline]
```

Construct start of input stream iterator.

References [std::\\_\\_addressof\(\)](#).

## 5.608.4 Friends And Related Symbol Documentation

### operator==

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
```

```

_Dist = ptrdiff_t>
bool operator== (
    const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x,
    const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y) [friend]

```

Return true if the iterators refer to the same stream, or are both at end-of-stream.

References [istream\\_iterator\(\)](#).

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 5.609 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference

```
#include <streambuf_iterator.h>
```

Inheritance diagram for std::istreambuf\_iterator< \_CharT, \_Traits >:



### Public Types

- typedef `_Traits::off_type` [difference\\_type](#)
- typedef `input_iterator_tag` [iterator\\_category](#)
- typedef `_CharT` [reference](#)
- typedef `_CharT` [value\\_type](#)
- using [pointer](#)
- typedef `_CharT` [char\\_type](#)
- typedef `_Traits` [traits\\_type](#)
- typedef `_Traits::int_type` [int\\_type](#)
- typedef `basic_streambuf< _CharT, _Traits >` [streambuf\\_type](#)
- typedef `basic_istream< _CharT, _Traits >` [istream\\_type](#)

### Public Member Functions

- constexpr [istreambuf\\_iterator](#) () noexcept
- [istreambuf\\_iterator](#) (const [istreambuf\\_iterator](#) &) noexcept=default



- `istreambuf_iterator` (`istream_type` &\_\_s) noexcept
- `istreambuf_iterator` (`streambuf_type` \*\_\_s) noexcept
- `bool equal` (const `istreambuf_iterator` &\_\_b) const
- `char_type operator*` () const
- `istreambuf_iterator` & `operator++` ()
- `istreambuf_iterator` `operator++` (int)
- `istreambuf_iterator` & `operator=` (const `istreambuf_iterator` &) noexcept=default

## Friends

- `template<bool _IsMove, typename _CharT2>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)`
- `template<typename _CharT2, typename _Size>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_n_a (istreambuf_iterator<`  
`_CharT2 >, _Size, _CharT2 *, bool)`
- `template<typename _CharT2, typename _Distance>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type advance (istreambuf_iterator< _`  
`CharT2 > &, _Distance)`
- `template<typename _CharT2>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 >::__type copy`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)`
- `template<typename _CharT2>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 >::__type find`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)`

### 5.609.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::istreambuf_iterator< _CharT, _Traits >
```

Provides input iterator semantics for streambufs.

### 5.609.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::istreambuf_iterator< _CharT, _Traits >::char_type
Public typedefs.
```

#### difference\_type

```
typedef _Traits::off_type std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *,
_CharT >::difference_type [inherited]
Distance between iterators is represented as this type.
```

#### int\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type
Public typedefs.
```

**istream\_type**

```
template<typename _CharT, typename _Traits>
typedef basic_istream<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::istream_type
Public typedefs.
```

**iterator\_category**

```
typedef input_iterator_tag std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *,
_CharT >::iterator_category [inherited]
One of the tag types.
```

**pointer**

```
template<typename _CharT, typename _Traits>
using std::istreambuf_iterator< _CharT, _Traits >::pointer
Public typedefs.
```

**reference**

```
typedef _CharT std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT >↵
::reference [inherited]
This type represents a reference-to-value_type.
```

**streambuf\_type**

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::streambuf↵
_type
Public typedefs.
```

**traits\_type**

```
template<typename _CharT, typename _Traits>
typedef _Traits std::istreambuf_iterator< _CharT, _Traits >::traits_type
Public typedefs.
```

**value\_type**

```
typedef _CharT std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT >↵
::value_type [inherited]
The type "pointed to" by the iterator.
```

**5.609.3 Constructor & Destructor Documentation****istreambuf\_iterator() [1/3]**

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator () [inline], [constexpr], [noexcept]
Construct end of input stream iterator.
Referenced by equal\(\), and operator++\(\).
```

**istreambuf\_iterator() [2/3]**

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
    istream_type & __s) [inline], [noexcept]
```

Construct start of input stream iterator.

#### **istreambuf\_iterator()** [3/3]

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
    streambuf_type * __s) [inline], [noexcept]
```

Construct start of streambuf iterator.

### **5.609.4 Member Function Documentation**

#### **equal()**

```
template<typename _CharT, typename _Traits>
bool std::istreambuf_iterator< _CharT, _Traits >::equal (
    const istreambuf_iterator< _CharT, _Traits > & __b) const [inline], [nodiscard]
```

Return true both iterators are end or both are not end.

References [istreambuf\\_iterator\(\)](#).

#### **operator\*()**

```
template<typename _CharT, typename _Traits>
char_type std::istreambuf_iterator< _CharT, _Traits >::operator* () const [inline], [nodiscard]
```

Return the current character pointed to by iterator. This returns streambuf.sgetc(). It cannot be assigned. NB: The result of operator\*() on an end of stream is undefined.

#### **operator++()** [1/2]

```
template<typename _CharT, typename _Traits>
istreambuf_iterator & std::istreambuf_iterator< _CharT, _Traits >::operator++ () [inline]
```

Advance the iterator. Calls streambuf.sbumpc().

#### **operator++()** [2/2]

```
template<typename _CharT, typename _Traits>
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits >::operator++ (
    int ) [inline]
```

Advance the iterator. Calls streambuf.sbumpc().

References [istreambuf\\_iterator\(\)](#).

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [streambuf\\_iterator.h](#)

### **5.610 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode<\_ATraits, Metadata >::iterator Struct Reference**

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator`:



## Public Types

- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value\_type**

## Public Member Functions

- **iterator** (`node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0`)
- bool **operator!=** (`const const_iterator &other`) const
- bool **operator!=** (`const iterator &other`) const
- `node_const_pointer` **operator\*** () const
- `node_pointer` **operator\*** ()
- `iterator &` **operator++** ()
- `iterator` **operator++** (int)
- `const node_pointer_pointer` **operator->** () const
- `node_pointer_pointer` **operator->** ()
- bool **operator==** (`const const_iterator &other`) const
- bool **operator==** (`const iterator &other`) const

## Public Attributes

- `node_pointer_pointer` **m\_p\_p\_cur**
- `node_pointer_pointer` **m\_p\_p\_end**

### 5.610.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata >::iterator
```

Child iterator.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.611 std::experimental::filesystem::v1::path::iterator Class Reference

```
#include <fs_path.h>
```

### Public Types

- using **difference\_type**
- using **iterator\_category**
- using **pointer**
- using **reference**
- using **value\_type**

### Public Member Functions

- **iterator** (const [iterator](#) &)=default
- **reference operator\*** () const noexcept
- **iterator & operator++** () noexcept
- **iterator operator++** (int) noexcept
- **iterator & operator--** () noexcept
- **iterator operator--** (int) noexcept
- **pointer operator->** () const noexcept
- **iterator & operator=** (const [iterator](#) &)=default

### Friends

- bool **operator!=** (const [iterator](#) &\_\_lhs, const [iterator](#) &\_\_rhs) noexcept
- bool **operator==** (const [iterator](#) &\_\_lhs, const [iterator](#) &\_\_rhs) noexcept
- class **path**

### 5.611.1 Detailed Description

An iterator for the components of a path.

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)

## 5.612 std::filesystem::path::iterator Class Reference

```
#include <filesystem>
```

### Public Types

- using **difference\_type**
- using **iterator\_category**
- using **pointer**
- using **reference**
- using **value\_type**

**Public Member Functions**

- `iterator` (const `iterator` &)=default
- `reference operator*` () const noexcept
- `iterator & operator++` () noexcept
- `iterator operator++` (int) noexcept
- `iterator & operator--` () noexcept
- `iterator operator--` (int) noexcept
- `pointer operator->` () const noexcept
- `iterator & operator=` (const `iterator` &)=default

**Friends**

- void `__path_iter_advance` (`iterator` &\_\_i, difference\_type \_\_n) noexcept
- difference\_type `__path_iter_distance` (const `iterator` &\_\_first, const `iterator` &\_\_last) noexcept
- bool `operator!=` (const `iterator` &\_\_lhs, const `iterator` &\_\_rhs) noexcept
- bool `operator==` (const `iterator` &\_\_lhs, const `iterator` &\_\_rhs) noexcept
- class `path`

**5.612.1 Detailed Description**

An iterator for the components of a path.

Since

C++17

The documentation for this class was generated from the following file:

- [bits/fs\\_path.h](#)

**5.613 `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >` Struct Template Reference**

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`:



## Public Types

- typedef `_Distance` [difference\\_type](#)
- typedef `_Category` [iterator\\_category](#)
- typedef `_Pointer` [pointer](#)
- typedef `_Reference` [reference](#)
- typedef `_Tp` [value\\_type](#)

### 5.613.1 Detailed Description

**template**<typename `_Category`, typename `_Tp`, typename `_Distance` = `ptrdiff_t`, typename `_Pointer` = `_Tp*`, typename `_Reference` = `_Tp&`>

**struct** `std::iterator`< `_Category`, `_Tp`, `_Distance`, `_Pointer`, `_Reference` >

Common iterator class.

This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

### 5.613.2 Member Typedef Documentation

#### `difference_type`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&>
typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_type
```

Distance between iterators is represented as this type.

#### `iterator_category`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&>
typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_category
```

One of the [tag types](#).

#### `pointer`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&>
typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer
```

This type represents a pointer-to-value\_type.

#### `reference`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&>
typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference
```

This type represents a reference-to-value\_type.

#### `value_type`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&>
typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type
```

The type "pointed to" by the iterator.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.614 `std::iterator_traits<_Iterator>` Struct Template Reference

### 5.614.1 Detailed Description

**template**<typename `_Iterator`>

**struct** `std::iterator_traits<_Iterator>`

Traits class for iterators.

This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the `Iterator` argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.615 `std::iterator_traits<_Tp*>` Struct Template Reference

```
#include <stl_iterator_base_types.h>
```

### Public Types

- using `difference_type`
- using `iterator_category`
- using `iterator_concept`
- using `pointer`
- using `reference`
- using `value_type`

### 5.615.1 Detailed Description

**template**<typename `_Tp`>

**requires** `is_object_v<_Tp>`

**struct** `std::iterator_traits<_Tp*>`

Partial specialization for object pointer types.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.616 `__gnu_pbds::join_error` Struct Reference

```
#include <exception.hpp>
```



Inheritance diagram for `__gnu_pbds::join_error`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 5.616.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.

#### 5.616.2 Member Function Documentation

##### what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

#### 5.617 `__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >` Class Template Reference

```
#include <left_child_next_sibling_heap_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `left_child_next_sibling_heap_const_iterator< node, _Alloc >` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `const_iterator` **iterator**
- typedef `left_child_next_sibling_heap_node< Value_Type, Node_Metadata, _Alloc >` **node**
- typedef `left_child_next_sibling_heap_node_point_const_iterator< node, _Alloc >` **point\_const\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- `left_child_next_sibling_heap` (`const Cmp_Fn &`)
- `left_child_next_sibling_heap` (`const left_child_next_sibling_heap &`)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `Cmp_Fn &` **get\_cmp\_fn** ()
- `const Cmp_Fn &` **get\_cmp\_fn** () const
- `size_type` **max\_size** () const
- `size_type` **size** () const
- void **swap** (`left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > &`)

## Protected Types

- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef alloc\_traits::const\_pointer **node\_const\_pointer**
- typedef Node\_Metadata **node\_metadata**
- typedef alloc\_traits::pointer **node\_pointer**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

## Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- template<typename Pred>  
node\_pointer **prune** (Pred)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

## Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

## Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

### 5.617.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >
```

Base class for a basic heap.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap.hpp](#)

### 5.618 `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` Class Template Reference

```
#include <const_iterator.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`:



## Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

## Public Member Functions

- `left_child_next_sibling_heap_const_iterator_()`
- `left_child_next_sibling_heap_const_iterator_(const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other)`
- `left_child_next_sibling_heap_const_iterator_(node_pointer p_nd)`
- `bool operator!= (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const`
- `bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`
- `const_reference operator* () const`
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & operator++ ()`
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc > operator++ (int)`
- `const_pointer operator-> () const`
- `bool operator== (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const`
- `bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`

## Public Attributes

- `node_pointer m_p_nd`

### 5.618.1 Detailed Description

`template<typename Node, typename _Alloc>`

`class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`

Const point-type iterator.

### 5.618.2 Member Typedef Documentation

#### `const_pointer`

`template<typename Node, typename _Alloc>`

typedef `base_type::const_pointer` `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_pointer`

Iterator's const pointer type.

#### `const_reference`

`template<typename Node, typename _Alloc>`

typedef `base_type::const_reference` `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_reference`

Iterator's const reference type.

**difference\_type**

```
template<typename Node, typename _Alloc>
typedef _Alloc::difference_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::difference_type
Difference type.
```

**iterator\_category**

```
template<typename Node, typename _Alloc>
typedef std::forward_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::iterator_category
Category.
```

**pointer**

```
template<typename Node, typename _Alloc>
typedef base_type::pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::pointer
Iterator's pointer type.
```

**reference**

```
template<typename Node, typename _Alloc>
typedef base_type::reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::reference
Iterator's reference type.
```

**value\_type**

```
template<typename Node, typename _Alloc>
typedef base_type::value_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::value_type
Iterator's value type.
```

**5.618.3 Constructor & Destructor Documentation****left\_child\_next\_sibling\_heap\_const\_iterator\_()** [1/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_↵
next_sibling_heap_const_iterator_ () [inline]
Default constructor.
```

**left\_child\_next\_sibling\_heap\_const\_iterator\_()** [2/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_↵
next_sibling_heap_const_iterator_ (
    const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) [inline]
Copy constructor.
```

**5.618.4 Member Function Documentation****operator"!="()** [1/2]

```
template<typename Node, typename _Alloc>
```

```
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator!=
(
    const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content (negatively) to a different iterator object.

#### **operator"!=()" [2/2]**

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

#### **operator\*()**

```
template<typename Node, typename _Alloc>
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const [inline], [inherited]
```

Access.

#### **operator->()**

```
template<typename Node, typename _Alloc>
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const [inline], [inherited]
```

Access.

#### **operator==(1/2)**

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator==
(
    const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content to a different iterator object.

#### **operator==(2/2)**

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const [inline], [inherited]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/const\\_iterator.hpp](#)

## 5.619 `__gnu_pbds::detail::left_child_next_sibling_heap_node_<_Value,_Metadata,_Alloc>` Struct Template Reference

```
#include <node.hpp>
```

## Public Types

- typedef `_Metadata` **metadata\_type**
- typedef `rebind_traits<_Alloc, this_type>::pointer` **node\_pointer**
- typedef `_Alloc::size_type` **size\_type**
- typedef `_Value` **value\_type**

## Public Attributes

- `metadata_type` **m\_metadata**
- `node_pointer` **m\_p\_l\_child**
- `node_pointer` **m\_p\_next\_sibling**
- `node_pointer` **m\_p\_prev\_or\_parent**
- `value_type` **m\_value**

### 5.619.1 Detailed Description

```
template<typename _Value, typename _Metadata, typename _Alloc>
struct __gnu_pbds::detail::left_child_next_sibling_heap_node<_Value, _Metadata, _Alloc>
```

Node.

The documentation for this struct was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/node.hpp](#)

### 5.620 `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator<Node, _Alloc>` Class Template Reference

```
#include <point_const_iterator.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator<Node, _Alloc>`:



## Public Types

- typedef `rebind_traits< _Alloc, value_type >::const_pointer` `const_pointer`
- typedef `rebind_traits< _Alloc, value_type >::const_reference` `const_reference`
- typedef `trivial_iterator_difference_type` `difference_type`
- typedef `trivial_iterator_tag` `iterator_category`
- typedef `rebind_traits< _Alloc, value_type >::pointer` `pointer`
- typedef `rebind_traits< _Alloc, value_type >::reference` `reference`
- typedef `Node::value_type` `value_type`

## Public Member Functions

- `left_child_next_sibling_heap_node_point_const_iterator_()`
- `left_child_next_sibling_heap_node_point_const_iterator_(const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other)`
- `left_child_next_sibling_heap_node_point_const_iterator_(node_pointer p_nd)`
- `bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`
- `const_reference operator* () const`
- `const_pointer operator-> () const`
- `bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`

## Public Attributes

- node\_pointer `m_p_nd`

## Protected Types

- typedef `rebind_traits< _Alloc, Node >::pointer` `node_pointer`

### 5.620.1 Detailed Description

`template<typename Node, typename _Alloc>`

`class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >`

Const point-type iterator.

### 5.620.2 Member Typedef Documentation

#### `const_pointer`

`template<typename Node, typename _Alloc>`

typedef `rebind_traits<_Alloc, value_type>::const_pointer` `__gnu_pbds::detail::left_child_next_sibling_heap_node_po`  
`Node, _Alloc >::const_pointer`

Iterator's const pointer type.

#### `const_reference`

`template<typename Node, typename _Alloc>`

typedef `rebind_traits<_Alloc, value_type>::const_reference` `__gnu_pbds::detail::left_child_next_sibling_heap_node_p`  
`Node, _Alloc >::const_reference`

Iterator's const reference type.



**difference\_type**

```
template<typename Node, typename _Alloc>
typedef trivial\_iterator\_difference\_type \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_
Node, _Alloc >::difference_type
Difference type.
```

**iterator\_category**

```
template<typename Node, typename _Alloc>
typedef trivial\_iterator\_tag \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_
Node, _Alloc >::iterator_category
Category.
```

**pointer**

```
template<typename Node, typename _Alloc>
typedef rebind\_traits<_Alloc, value\_type>::pointer \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_
Node, _Alloc >::pointer
Iterator's pointer type.
```

**reference**

```
template<typename Node, typename _Alloc>
typedef rebind\_traits<_Alloc, value\_type>::reference \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_
Node, _Alloc >::reference
Iterator's reference type.
```

**value\_type**

```
template<typename Node, typename _Alloc>
typedef Node::value_type \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_
Node, _Alloc >::value_type
Iterator's value type.
```

**5.620.3 Constructor & Destructor Documentation****`left_child_next_sibling_heap_node_point_const_iterator_()` [1/2]**

```
template<typename Node, typename _Alloc>
\_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, _Alloc >↔
::left_child_next_sibling_heap_node_point_const_iterator_ () [inline]
Default constructor.
```

**`left_child_next_sibling_heap_node_point_const_iterator_()` [2/2]**

```
template<typename Node, typename _Alloc>
\_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, _Alloc >↔
::left_child_next_sibling_heap_node_point_const_iterator_ (
    const left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, _Alloc > &
    other) [inline]
Copy constructor.
```

#### 5.620.4 Member Function Documentation

##### operator!=(())

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
    other) const [inline]
```

Compares content (negatively) to a different iterator object.

##### operator\*()

```
template<typename Node, typename _Alloc>
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const [inline]
```

Access.

##### operator->()

```
template<typename Node, typename _Alloc>
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const [inline]
```

Access.

##### operator==(())

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
    other) const [inline]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/point\\_const\\_iterator.hpp](#)

## 5.621 std::length\_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for `std::length_error`:



### Public Member Functions

- `length_error` (const char \*)
- `length_error` (const [length\\_error](#) &)=default
- `length_error` (const [string](#) &\_\_arg)
- `length_error` ([length\\_error](#) &&)=default
- `length_error` & `operator=` (const [length\\_error](#) &)=default
- `length_error` & `operator=` ([length\\_error](#) &&)=default
- virtual const char \* `what` () const noexcept

#### 5.621.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a `basic_string` instance).

#### 5.621.2 Member Function Documentation

##### `what()`

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 5.622 `std::less<_Tp>` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::less< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 5.622.1 Detailed Description

**template<typename \_Tp>**  
**struct std::less< \_Tp >**

One of the [comparison functors](#).

#### 5.622.2 Member Typedef Documentation

##### **first\_argument\_type**

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Tp`, `bool` >::`first_argument_type` [inherited]  
`first_argument_type` is the type of the first argument

##### **result\_type**

typedef `bool` [std::binary\\_function](#)< `_Tp`, `_Tp`, `bool` >::`result_type` [inherited]  
`result_type` is the return type

##### **second\_argument\_type**

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Tp`, `bool` >::`second_argument_type` [inherited]  
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.623 std::ranges::less Struct Reference

```
#include <ranges_cmp.h>
```

### Public Types

- using `is_transparent`

### Public Member Functions

- `template<typename _Tp, typename _Up>`  
requires `totally_ordered_with<_Tp, _Up>`  
`constexpr bool operator() (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval<_Tp>())<  
std::declval<_Up>()))`

#### 5.623.1 Detailed Description

`ranges::less` function object type.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 5.624 std::less< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::less< void >`:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` `is_transparent`
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- `template<typename _Tp, typename _Up>`  
`constexpr auto operator() (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward<_Tp>(__t)<  
std::forward<_Up>(__u))) -> decltype(std::forward<_Tp>(__t)< std::forward<_Up>(__u))`

- `template<typename _Tp, typename _Up>`  
`constexpr bool operator() (_Tp *__t, _Up *__u) const noexcept`
- `constexpr bool operator() (const void &__x, const void &__y) const`

### 5.624.1 Detailed Description

One of the [comparison functors](#).

### 5.624.2 Member Typedef Documentation

#### first\_argument\_type

`typedef void std::binary\_function< void, void, bool >::first_argument_type`  
`first_argument_type` is the type of the first argument

#### result\_type

`typedef bool std::binary\_function< void, void, bool >::result_type`  
`result_type` is the return type

#### second\_argument\_type

`typedef void std::binary\_function< void, void, bool >::second_argument_type`  
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.625 std::less\_equal< \_Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::less_equal< _Tp >`:



### Public Types

- `typedef _Tp first\_argument\_type`
- `typedef bool result\_type`
- `typedef _Tp second\_argument\_type`

## Public Member Functions

- constexpr bool **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

### 5.625.1 Detailed Description

```
template<typename _Tp>
struct std::less_equal<_Tp >
```

One of the [comparison functors](#).

### 5.625.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.626 std::ranges::less\_equal Struct Reference

```
#include <ranges_cmp.h>
```

## Public Types

- using **is\_transparent**

## Public Member Functions

- template<typename \_Tp, typename \_Up>  
requires totally\_ordered\_with<\_Tp, \_Up>  
constexpr bool **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::declval](#)< \_Up >())<  
[std::declval](#)< \_Tp >()))

### 5.626.1 Detailed Description

ranges::less\_equal function object type.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 5.627 std::less\_equal< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::less\_equal< void >:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef \_\_is\_transparent [is\\_transparent](#)
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t) <= [std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t) <= [std::forward](#)< \_Up >(\_\_u))
- template<typename \_Tp, typename \_Up>  
constexpr bool **operator()** (\_Tp \*\_\_t, \_Up \*\_\_u) const noexcept
- constexpr bool **operator()** (const void &\_\_x, const void &\_\_y) const

#### 5.627.1 Detailed Description

One of the [comparison functors](#).

#### 5.627.2 Member Typedef Documentation

##### first\_argument\_type

```
typedef void std::binary\_function< void, void, bool >::first_argument_type
```

[first\\_argument\\_type](#) is the type of the first argument

##### result\_type

```
typedef bool std::binary\_function< void, void, bool >::result_type
```

[result\\_type](#) is the return type



## second\_argument\_type

typedef void [std::binary\\_function](#)< void, void, bool >::second\_argument\_type

second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.628 \_\_gnu\_cxx::limit\_condition::limit\_adjustor Struct Reference

```
#include <throw_allocator.h>
```

### Public Member Functions

- **limit\_adjustor** (const size\_t \_\_l)

#### 5.628.1 Detailed Description

Enter the nth condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.629 \_\_gnu\_cxx::limit\_condition Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::limit\_condition:



### Classes

- struct [always\\_adjustor](#)
- struct [limit\\_adjustor](#)
- struct [never\\_adjustor](#)

### Static Public Member Functions

- static size\_t & **count** ()
- static size\_t & **limit** ()
- static void **set\_limit** (const size\_t \_\_l)
- static void **throw\_conditionally** ()

#### 5.629.1 Detailed Description

Base class for incremental control and throw.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.630 std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m > Class Template Reference

```
#include <random>
```

### Public Types

- typedef \_UIntType [result\\_type](#)

### Public Member Functions

- [linear\\_congruential\\_engine](#) ()
- template<typename \_Sseq, typename = \_If\_seed\_seq<\_Sseq>>  
[linear\\_congruential\\_engine](#) (\_Sseq & \_\_q)
- [linear\\_congruential\\_engine](#) (result\_type \_\_s)
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) operator() ()
- template<typename \_Sseq>  
\_If\_seed\_seq<\_Sseq> > [seed](#) (\_Sseq & \_\_q)
- template<typename \_Sseq>  
auto [seed](#) (\_Sseq & \_\_q) -> \_If\_seed\_seq<\_Sseq>
- void [seed](#) (result\_type \_\_s=default\_seed)

### Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

### Static Public Attributes

- static constexpr [result\\_type](#) [default\\_seed](#)
- static constexpr [result\\_type](#) [increment](#)
- static constexpr [result\\_type](#) [modulus](#)
- static constexpr [result\\_type](#) [multiplier](#)

### Friends

- template<typename \_UIntType1, \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > & \_\_os, const  
[std::linear\\_congruential\\_engine](#)< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > & \_\_lcr)
- bool [operator==](#) (const [linear\\_congruential\\_engine](#) & \_\_lhs, const [linear\\_congruential\\_engine](#) & \_\_rhs)
- template<typename \_UIntType1, \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > & \_\_is,  
[std::linear\\_congruential\\_engine](#)< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > & \_\_lcr)

#### 5.630.1 Detailed Description

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
class std::linear_congruential_engine< _UIntType, __a, __c, __m >
```

A model of a linear congruential random number generator.

A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is `std::numeric_limits<_UIntType>::max()` plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Since

C++11

### 5.630.2 Member Typedef Documentation

#### result\_type

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
typedef _UIntType std::linear\_congruential\_engine< _UIntType, __a, __c, __m >::result_type
```

The type of the generated random value.

### 5.630.3 Constructor & Destructor Documentation

#### linear\_congruential\_engine() [1/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear\_congruential\_engine< _UIntType, __a, __c, __m >::linear_congruential_engine () [inline]
```

Constructs a `linear_congruential_engine` random number generator engine with seed 1.

#### linear\_congruential\_engine() [2/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear\_congruential\_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
    result_type __s) [inline], [explicit]
```

Constructs a `linear_congruential_engine` random number generator engine with seed `__s`. The default seed value is 1.

##### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The initial seed value. |
|------------------|-------------------------|

#### linear\_congruential\_engine() [3/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::linear\_congruential\_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
    _Sseq & __q) [inline], [explicit]
```

Constructs a `linear_congruential_engine` random number generator engine seeded from the seed sequence `__q`.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

#### 5.630.4 Member Function Documentation

##### discard()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine< _UIntType, __a, __c, __m >::discard (
    unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

##### max()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
static constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::max ()
[inline], [static], [constexpr]
```

Gets the largest possible value in the output range.

##### min()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
static constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::min ()
[inline], [static], [constexpr]
```

Gets the smallest possible value in the output range.

The minimum depends on the \_\_c parameter: if it is zero, the minimum generated must be > 0, otherwise 0 is allowed.

##### operator()()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::operator() () [inline]
```

Gets the next random number in the sequence.

##### seed() [1/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
    _Sseq & __q)
```

Reseeds the linear\_congruential\_engine random number generator engine sequence using values from the seed sequence \_\_q.

##### Parameters

|                   |                    |
|-------------------|--------------------|
| $\leftrightarrow$ | the seed sequence. |
| <code>__q</code>  |                    |

##### seed() [2/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq>
auto std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
    _Sseq & __q) -> _If_seed_seq<_Sseq>
```

Seeds the LCR engine with a value generated by \_\_q.

**seed()** [3/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
    result_type __s = default_seed)
```

Reseeds the linear\_congruential\_engine random number generator engine sequence to the seed \_\_s.

**Parameters**

|                  |               |
|------------------|---------------|
| <code>__s</code> | The new seed. |
|------------------|---------------|

Seeds the LCR with integral value \_\_s, adjusted so that the ring identity is never a member of the convergence set.

Referenced by `std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::linear_congruential_engine()`, and `std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::linear_congruential_engine()`.

**5.630.5 Friends And Related Symbol Documentation****operator<<**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT,
typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr) [friend]
```

Writes the textual representation of the state x(i) of x to \_\_os.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__os</code>  | The output stream.                                      |
| <code>__lcr</code> | A % linear_congruential_engine random number generator. |

**Returns**

`__os`.

**operator==**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool operator== (
    const linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs,
    const linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs) [friend]
```

Compares two linear congruential random number generator objects of the same type for equality.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__lhs</code> | A linear congruential random number generator object.       |
| <code>__rhs</code> | Another linear congruential random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

**operator>>**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT,
typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr) [friend]
```

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

**Parameters**

|                    |                                                                      |
|--------------------|----------------------------------------------------------------------|
| <code>__is</code>  | The input stream.                                                    |
| <code>__lcr</code> | A % <code>linear_congruential_engine</code> random number generator. |

**Returns**

`__is`.

**5.630.6 Member Data Documentation****increment**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::increment [static],
[constexpr]
```

An increment.

**modulus**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::modulus [static], [constexpr]
```

The modulus.

**multiplier**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::multiplier [static],
[constexpr]
```

The multiplier.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.631 `__gnu_pbds::linear_probe_fn< Size_Type >` Class Template Reference**

```
#include <hash_policy.hpp>
```

**Public Types**

- typedef `Size_Type` **size\_type**

## Public Member Functions

- void **swap** ([linear\\_probe\\_fn](#)< Size\_Type > &other)

## Protected Member Functions

- size\_type [operator\(\)](#) (size\_type i) const

### 5.631.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::linear_probe_fn< Size_Type >
```

A probe sequence policy using fixed increments.

### 5.631.2 Member Function Documentation

#### **operator()**

```
template<typename Size_Type>
linear\_probe\_fn< Size_Type >::size_type \_\_gnu\_pbds::linear\_probe\_fn< Size_Type >::operator() (
    size_type i) const [inline], [protected]
```

Returns the i-th offset from the hash value.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 5.632 std::\_\_debug::list< \_Tp, \_Allocator > Class Template Reference

```
#include <list>
```

Inheritance diagram for std::\_\_debug::list< \_Tp, \_Allocator >:



## Public Types

- typedef \_Allocator **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< [\\_Base\\_const\\_iterator](#), list > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< [const\\_iterator](#) > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< [\\_Base\\_iterator](#), list > **iterator**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**

- typedef [std::reverse\\_iterator](#)< [iterator](#) > **reverse\_iterator**
- typedef [\\_Base::size\\_type](#) **size\_type**
- typedef [\\_Tp](#) **value\_type**

## Public Member Functions

- **list** ([\\_Base\\_ref](#) \_\_x)
- template<class [\\_InputIterator](#), typename = [std::\\_RequireInputIter](#)<[\\_InputIterator](#)>>  
**list** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **list** (const [\\_Allocator](#) &\_\_a) noexcept
- **list** (const [list](#) &)=default
- **list** (const [list](#) &\_\_x, const [\\_\\_type\\_identity\\_t](#)< [allocator\\_type](#) > &\_\_a)
- **list** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **list** ([list](#) &&)=default
- **list** ([list](#) &&\_\_x, const [\\_\\_type\\_identity\\_t](#)< [allocator\\_type](#) > &\_\_a) noexcept([std::is\\_nothrow\\_constructible](#)<[\\_Base](#), [\\_Base](#), const [allocator\\_type](#) & >::value)
- **list** ([size\\_type](#) \_\_n, const [\\_\\_type\\_identity\\_t](#)< [\\_Tp](#) > &\_\_value, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **list** ([size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- const [\\_Base](#) & **\_M\_base** () const noexcept
- [\\_Base](#) & **\_M\_base** () noexcept
- template<class [\\_InputIterator](#), typename = [std::\\_RequireInputIter](#)<[\\_InputIterator](#)>>  
void **assign** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- void **assign** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- void **assign** ([size\\_type](#) \_\_n, const [\\_Tp](#) &\_\_t)
- const\_reference **back** () const noexcept
- reference **back** () noexcept
- const\_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... [\\_Args](#)>  
[iterator](#) **emplace** (const\_iterator \_\_position, [\\_Args](#) &&... \_\_args)
- const\_iterator **end** () const noexcept
- iterator **end** () noexcept
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- iterator **erase** (const\_iterator \_\_position) noexcept
- const\_reference **front** () const noexcept
- reference **front** () noexcept
- iterator **insert** (const\_iterator \_\_p, [initializer\\_list](#)< [value\\_type](#) > \_\_l)
- template<class [\\_InputIterator](#), typename = [std::\\_RequireInputIter](#)<[\\_InputIterator](#)>>  
[iterator](#) **insert** (const\_iterator \_\_position, [\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- iterator **insert** (const\_iterator \_\_position, [\\_Tp](#) &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, const [\\_Tp](#) &\_\_x)
- iterator **insert** (const\_iterator \_\_position, [size\\_type](#) \_\_n, const [\\_Tp](#) &\_\_x)
- void **merge** ([list](#) &&\_\_x)
- template<class [\\_Compare](#)>  
void **merge** ([list](#) &&\_\_x, [\\_Compare](#) \_\_comp)
- void **merge** ([list](#) &\_\_x)



- `template<typename _Compare>`  
`void merge (list &__x, _Compare __comp)`
- `list & operator= (const list &)=default`
- `list & operator= (initializer_list< value_type > __l)`
- `list & operator= (list &&)=default`
- `void pop_back () noexcept`
- `void pop_front () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rbegin () noexcept`
- `__remove_return_type remove (const _Tp &__value)`
- `template<class _Predicate>`  
`__remove_return_type remove_if (_Predicate __pred)`
- `const_reverse_iterator rend () const noexcept`
- `reverse_iterator rend () noexcept`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const _Tp &__c)`
- `void sort ()`
- `template<typename _StrictWeakOrdering>`  
`void sort (_StrictWeakOrdering __pred)`
- `void splice (const_iterator __position, list &&__x) noexcept`
- `void splice (const_iterator __position, list &&__x, const_iterator __first, const_iterator __last) noexcept`
- `void splice (const_iterator __position, list &&__x, const_iterator __i) noexcept`
- `void splice (const_iterator __position, list &__x) noexcept`
- `void splice (const_iterator __position, list &__x, const_iterator __first, const_iterator __last) noexcept`
- `void splice (const_iterator __position, list &__x, const_iterator __i) noexcept`
- `void swap (list &__x) noexcept(/*conditional */)`
- `__remove_return_type unique ()`
- `template<class _BinaryPredicate>`  
`__remove_return_type unique (_BinaryPredicate __binary_pred)`

### Protected Member Functions

- `constexpr void _M_swap (const _Safe_container &__x) const noexcept`

### Friends

- `template<typename _ItT, typename _SeqT, typename _CatT>`  
`class ::__gnu_debug::Safe_iterator`

### 5.632.1 Detailed Description

`template<typename _Tp, typename _Allocator = std::allocator<_Tp>>`  
`class std::__debug::list< _Tp, _Allocator >`

Class `std::list` with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [debug/list](#)

## 5.633 std::list< \_Tp, \_Alloc > Class Template Reference

```
#include <stl_list.h>
```

Inheritance diagram for std::list< \_Tp, \_Alloc >:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Node_traits::_Const_iterator` **const\_iterator**
- typedef `_Tp_alloc_traits::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Node_traits::_Iterator` **iterator**
- typedef `_Tp_alloc_traits::pointer` **pointer**
- typedef `_Tp_alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- `list()`=default
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
`list` (`_InputIterator` `__first`, `_InputIterator` `__last`, `const allocator_type &__a`=`allocator_type()`)
- `list` (`const allocator_type &__a`) noexcept
- `list` (`const list &__x`)
- `list` (`const list &__x`, `const __type_identity_t< allocator_type > &__a`)
- `list` (`initializer_list< value_type > __l`, `const allocator_type &__a`=`allocator_type()`)
- `list` (`list &&`)=default
- `list` (`list &&__x`, `const __type_identity_t< allocator_type > &__a`) noexcept(`_Node_alloc_traits::_S_always_equal()`)
- `list` (`size_type __n`, `const allocator_type &__a`=`allocator_type()`)
- `list` (`size_type __n`, `const value_type &__value`, `const allocator_type &__a`=`allocator_type()`)
- `~list()`=default
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
`void assign` (`_InputIterator` `__first`, `_InputIterator` `__last`)

- void `assign` (`initializer_list`< `value_type` > \_\_l)
- void `assign` (`size_type` \_\_n, const `value_type` &\_\_val)
- const\_reference `back` () const noexcept
- reference `back` () noexcept
- const\_iterator `begin` () const noexcept
- iterator `begin` () noexcept
- const\_iterator `cbegin` () const noexcept
- const\_iterator `cend` () const noexcept
- void `clear` () noexcept
- const\_reverse\_iterator `crbegin` () const noexcept
- const\_reverse\_iterator `crend` () const noexcept
- template<typename... \_\_Args>  
iterator `emplace` (const\_iterator \_\_position, \_\_Args &&... \_\_args)
- template<typename... \_\_Args>  
reference `emplace_back` (\_\_Args &&... \_\_args)
- template<typename... \_\_Args>  
reference `emplace_front` (\_\_Args &&... \_\_args)
- bool `empty` () const noexcept
- const\_iterator `end` () const noexcept
- iterator `end` () noexcept
- iterator `erase` (const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- iterator `erase` (const\_iterator \_\_position) noexcept
- const\_reference `front` () const noexcept
- reference `front` () noexcept
- allocator\_type `get_allocator` () const noexcept
- iterator `insert` (const\_iterator \_\_p, `initializer_list`< `value_type` > \_\_l)
- template<typename \_\_InputIterator, typename = std::\_\_RequireInputIter<\_\_InputIterator>>  
iterator `insert` (const\_iterator \_\_position, \_\_InputIterator \_\_first, \_\_InputIterator \_\_last)
- iterator `insert` (const\_iterator \_\_position, `size_type` \_\_n, const `value_type` &\_\_x)
- `size_type` `max_size` () const noexcept
- void `merge` (`list` &&\_\_x)
- template<typename \_\_StrictWeakOrdering>  
void `merge` (`list` &&\_\_x, \_\_StrictWeakOrdering \_\_comp)
- void `merge` (`list` &\_\_x)
- template<typename \_\_StrictWeakOrdering>  
void `merge` (`list` &\_\_x, \_\_StrictWeakOrdering \_\_comp)
- `list` & `operator=` (const `list` &\_\_x)
- `list` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- `list` & `operator=` (`list` &&\_\_x) noexcept(`_Node_alloc_traits::_S_nothrow_move`())
- void `pop_back` () noexcept
- void `pop_front` () noexcept
- void `push_back` (const `value_type` &\_\_x)
- void `push_back` (`value_type` &&\_\_x)
- void `push_front` (const `value_type` &\_\_x)
- void `push_front` (`value_type` &&\_\_x)
- const\_reverse\_iterator `rbegin` () const noexcept
- reverse\_iterator `rbegin` () noexcept
- \_\_remove\_return\_type `remove` (const \_\_Tp &\_\_value)
- template<typename \_\_Predicate>  
\_\_remove\_return\_type `remove_if` (\_\_Predicate)
- const\_reverse\_iterator `rend` () const noexcept

- [reverse\\_iterator rend](#) () noexcept
- void [resize](#) (size\_type \_\_new\_size)
- void [resize](#) (size\_type \_\_new\_size, const value\_type &\_\_x)
- void [reverse](#) () noexcept
- size\_type [size](#) () const noexcept
- void [sort](#) ()
- template<typename \_StrictWeakOrdering>  
void [sort](#) (\_StrictWeakOrdering)
- void [splice](#) (const\_iterator \_\_position, list &&\_\_x) noexcept
- void [splice](#) (const\_iterator \_\_position, list &&\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void [splice](#) (const\_iterator \_\_position, list &\_\_x) noexcept
- void [splice](#) (const\_iterator \_\_position, list &\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void [swap](#) (list &\_\_x) noexcept
- \_\_remove\_return\_type [unique](#) ()
- template<typename \_BinaryPredicate>  
\_\_remove\_return\_type [unique](#) (\_BinaryPredicate)
- iterator [insert](#) (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator [insert](#) (const\_iterator \_\_position, value\_type &&\_\_x)
- void [splice](#) (const\_iterator \_\_position, list &&\_\_x, const\_iterator \_\_i) noexcept
- void [splice](#) (const\_iterator \_\_position, list &\_\_x, const\_iterator \_\_i) noexcept

### Protected Types

- typedef \_Node\_alloc\_traits::pointer **\_Node\_ptr**

### Protected Member Functions

- template<typename \_InputIterator>  
void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_Integer>  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- void **\_M\_check\_equal\_allocators** (const list &\_\_x) noexcept
- void **\_M\_clear** () noexcept
- template<typename... \_Args>  
\_Node\_ptr **\_M\_create\_node** (\_Args &&... \_\_args)
- void **\_M\_dec\_size** (size\_t \_\_n)
- void **\_M\_default\_append** (size\_type \_\_n)
- void **\_M\_default\_initialize** (size\_type \_\_n)
- void **\_M\_destroy\_node** (\_Node\_ptr \_\_p)
- size\_t **\_M\_distance** (const \_\_detail::List\_node\_base \* \_\_first, const \_\_detail::List\_node\_base \* \_\_last) const
- void **\_M\_erase** (iterator \_\_position) noexcept
- void **\_M\_fill\_assign** (size\_type \_\_n, const value\_type &\_\_val)
- void **\_M\_fill\_initialize** (size\_type \_\_n, const value\_type &\_\_x)
- \_Node\_alloc\_traits::pointer **\_M\_get\_node** ()
- const \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () const noexcept
- \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () noexcept
- size\_t **\_M\_get\_size** () const
- void **\_M\_inc\_size** (size\_t \_\_n)

- void **\_M\_init** () noexcept
- template<typename \_InputIterator>  
void **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_Integer>  
void **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_x, \_\_true\_type)
- template<typename... \_Args>  
void **\_M\_insert** (iterator \_\_position, \_Args &&... \_\_args)
- void **\_M\_move\_assign** (list &&\_\_x, false\_type)
- void **\_M\_move\_assign** (list &&\_\_x, true\_type) noexcept
- void **\_M\_move\_nodes** (\_List\_base &&\_\_x)
- size\_t **\_M\_node\_count** () const
- void **\_M\_put\_node** (\_Node\_ptr \_\_p) noexcept
- const\_iterator **\_M\_resize\_pos** (size\_type &\_\_new\_size) const
- void **\_M\_set\_size** (size\_t \_\_n)
- void **\_M\_transfer** (iterator \_\_position, iterator \_\_first, iterator \_\_last)

### Static Protected Member Functions

- static size\_t **\_S\_distance** (const \_\_detail::\_List\_node\_base \*\_\_first, const \_\_detail::\_List\_node\_base \*\_\_last)
- static size\_t **\_S\_distance** (const\_iterator \_\_first, const\_iterator \_\_last)

### Protected Attributes

- \_List\_impl \_M\_impl

### 5.633.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::list< _Tp, _Alloc >
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X,Alloc1>` are spliced into `list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

```
A <----> B <----> C <----> D
```

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private `list::iterator` pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's `next/previous` pointers refer to itself, the list is empty.

### 5.633.2 Constructor & Destructor Documentation

#### list() [1/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list () [default]
```

Creates a list with no elements.

Referenced by [erase\(\)](#), [insert\(\)](#), [insert\(\)](#), [insert\(\)](#), [merge\(\)](#), [merge\(\)](#), [operator=\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), [unique\(\)](#), and [unique\(\)](#).

#### list() [2/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list (
    const allocator_type & __a) [inline], [explicit], [noexcept]
```

Creates a list with no elements.

##### Parameters

|                                                                                                                       |                      |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__a</code> | An allocator object. |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|

#### list() [3/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list (
    size_type __n,
    const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a list with default constructed elements.

##### Parameters

|                                                                                                                         |                                             |
|-------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| <a href="#"></a><br><code>__n</code> | The number of elements to initially create. |
| <a href="#"></a><br><code>__a</code> | An allocator object.                        |

This constructor fills the list with `__n` default constructed elements.

#### list() [4/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list (
    size_type __n,
    const value_type & __value,
    const allocator_type & __a = allocator_type()) [inline]
```

Creates a list with copies of an exemplar element.

##### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator object.                        |

This constructor fills the list with `__n` copies of `__value`.

**list()** [5/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
    const list< _Tp, _Alloc > & __x) [inline]
```

List copy constructor.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

The newly-created list uses a copy of the allocation object used by `__x` (unless the allocator traits dictate a different object).

**list()** [6/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
    list< _Tp, _Alloc > && ) [default]
```

List move constructor.

The newly-created list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified list.

**list()** [7/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
    initializer_list< value_type > __l,
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a list from an `initializer_list`.

**Parameters**

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> of <code>value_type</code> . |
| <code>__a</code> | An allocator object.                                          |

Create a list consisting of copies of the elements in the `initializer_list __l`. This is linear in `__l.size()`.

**list()** [8/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::list< _Tp, _Alloc >::list (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a list from a range.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

Create a list consisting of copies of the elements from `[__first,__last)`. This is linear in N (where N is distance(`__first`,`__last`)).

### `~list()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::~~list () [default]
```

No explicit dtor needed as the `_Base` dtor takes care of things. The `_Base` dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 5.633.3 Member Function Documentation

### `_M_create_node()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename... _Args>
_Node_ptr std::list< _Tp, _Alloc >::_M_create_node (
    _Args &&... __args) [inline], [protected]
```

#### Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>__args</code> | An instance of user data. |
|---------------------|---------------------------|

Allocates space for a new node and constructs a copy of `__args` in it.  
Referenced by [emplace\(\)](#), and [insert\(\)](#).

### `assign()` [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::list< _Tp, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Assigns a range to a list.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a list with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

### `assign()` [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (
    initializer_list< value_type > __l) [inline]
```

Assigns an `initializer_list` to a list.



**Parameters**

|   |                                    |
|---|------------------------------------|
| ↩ | An initializer_list of value_type. |
| ↩ |                                    |
| ↩ |                                    |
| ↩ |                                    |
| / |                                    |

Replace the contents of the list with copies of the elements in the initializer\_list \_\_l. This is linear in \_\_l.size().

**assign()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::assign (
    size_type __n,
    const value_type & __val) [inline]
```

Assigns a given value to a list.

**Parameters**

|       |                                    |
|-------|------------------------------------|
| __n   | Number of elements to be assigned. |
| __val | Value to be assigned.              |

This function fills a list with \_\_n copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

**back()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list<_Tp, _Alloc>::back () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the list.

**back()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list<_Tp, _Alloc>::back () [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the last element of the list.

**begin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list<_Tp, _Alloc>::begin () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

**begin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list<_Tp, _Alloc>::begin () [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order. Referenced by `std::list<_Tp, polymorphic_allocator<_Tp>>::list()`, `insert()`, `insert()`, `merge()`, `merge()`, `std::operator<=>()`, `operator=()`, `std::list<_Tp, polymorphic_allocator<_Tp>>::operator=()`, `remove()`, `remove_if()`, `std::list<_Tp, polymorphic_allocator<_Tp>>::unique()`, and `unique()`.

**cbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list<_Tp, _Alloc>::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

**cend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list<_Tp, _Alloc>::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

**clear()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::clear () [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility. Referenced by [operator=\(\)](#).

**crbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list<_Tp, _Alloc>::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

**crend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list<_Tp, _Alloc>::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

**emplace()**

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
list<_Tp, _Alloc>::iterator list::emplace (
    const_iterator __position,
    _Args &&... __args)
```

Constructs object in list before specified iterator.

**Parameters**

|                         |                                              |
|-------------------------|----------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the list. |
| <code>__args</code>     | Arguments.                                   |

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references. References \_M\_create\_node\(\), and std::forward\(\).`

**empty()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::list< _Tp, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the list is empty. (Thus `begin()` would equal `end()`.)

Referenced by [insert\(\)](#), and [std::list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::splice\(\)](#).

**end()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::end () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

**end()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::end () [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Referenced by [std::list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::list\(\)](#), [merge\(\)](#), [merge\(\)](#), [std::operator<=>\(\)](#), [operator=\(\)](#), [std::list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::operator=\(\)](#), and [std::list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::splice\(\)](#).

**erase()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last) [inline], [noexcept]
```

Remove a range of elements.

**Parameters**

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

**Returns**

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/2]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::erase (
    const_iterator __position) [noexcept]
```

Remove element at given position.

**Parameters**

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

**Returns**

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), and [erase\(\)](#).

Referenced by [erase\(\)](#), [resize\(\)](#), and [resize\(\)](#).

**front()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list<_Tp, _Alloc>::front () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the list.

**front()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list<_Tp, _Alloc>::front () [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the first element of the list.

**get\_allocator()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::list<_Tp, _Alloc>::get_allocator () const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Referenced by [insert\(\)](#), [insert\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), [unique\(\)](#), and [unique\(\)](#).

**insert()** [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list<_Tp, _Alloc>::insert (
    const_iterator __p,
    initializer_list<value_type> __l) [inline]
```

Inserts the contents of an `initializer_list` into list before specified `const_iterator`.

**Parameters**

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| <code>__p</code> | A <code>const_iterator</code> into the list.                  |
| <code>__l</code> | An <code>initializer_list</code> of <code>value_type</code> . |

**Returns**

An iterator pointing to the first element inserted (or `__p`).

This function will insert copies of the data in the `initializer_list __l` into the list before the location specified by `__p`. This operation is linear in the number of elements inserted and does not invalidate iterators and references.

**insert()** [2/5]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator, typename>
```

```
list< _Tp, _Alloc >::iterator list::insert (
    const_iterator __position,
    _InputIterator __first,
    _InputIterator __last)
```

Inserts a range into the list.

#### Parameters

|                         |                            |
|-------------------------|----------------------------|
| <code>__position</code> | An iterator into the list. |
| <code>__first</code>    | An input iterator.         |
| <code>__last</code>     | An input iterator.         |

#### Returns

Since C++11, an iterator pointing to the first element inserted (or `__position`). In C++98 this returns void.

This function will insert copies of the data in the range `[__first, __last)` into the list before the location specified by `__position`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

References [list\(\)](#), [begin\(\)](#), [empty\(\)](#), [get\\_allocator\(\)](#), and [splice\(\)](#).

#### insert() [3/5]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::insert (
    const_iterator __position,
    const value_type & __x)
```

Inserts given value into list before specified iterator.

#### Parameters

|                         |                            |
|-------------------------|----------------------------|
| <code>__position</code> | An iterator into the list. |
| <code>__x</code>        | Data to be inserted.       |

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

References [list\(\)](#), [\\_M\\_create\\_node\(\)](#), and [insert\(\)](#).

Referenced by [insert\(\)](#), and [resize\(\)](#).

#### insert() [4/5]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::insert (
    const_iterator __position,
    size_type __n,
    const value_type & __x)
```

Inserts a number of copies of given data into the list.

## Parameters

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | An iterator into the list.         |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

## Returns

Since C++11, an iterator pointing to the first element inserted (or `__position`). In C++98 this returns void.

This function will insert a specified number of copies of the given data before the location specified by `__position`. This operation is linear in the number of elements inserted and does not invalidate iterators and references. References [list\(\)](#), [begin\(\)](#), [get\\_allocator\(\)](#), and [splice\(\)](#).

**insert()** [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list<_Tp, _Alloc>::insert (
    const_iterator __position,
    value_type && __x) [inline]
```

Inserts given value into list before specified iterator.

## Parameters

|                         |                            |
|-------------------------|----------------------------|
| <code>__position</code> | An iterator into the list. |
| <code>__x</code>        | Data to be inserted.       |

## Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

**max\_size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list<_Tp, _Alloc>::max_size () const [inline], [nodiscard], [noexcept]
```

Returns the size() of the largest possible list.

**merge()** [1/2]

```
template<typename _Tp, typename _Alloc>
void list::merge (
    list<_Tp, _Alloc> && __x)
```

Merge sorted lists.

## Parameters

|                  |                       |
|------------------|-----------------------|
| <code>__x</code> | Sorted list to merge. |
|------------------|-----------------------|

Assumes that both `__x` and this list are sorted according to operator<(). Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equal.

References [list\(\)](#), [std::\\_\\_addressof\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [end\(\)](#), and [merge\(\)](#).

Referenced by [merge\(\)](#), and [merge\(\)](#).

**merge()** [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _StrictWeakOrdering>
void list::merge (
    list< _Tp, _Alloc > && __x,
    _StrictWeakOrdering __comp)
```

Merge sorted lists according to comparison function.

**Template Parameters**

|                                  |                                          |
|----------------------------------|------------------------------------------|
| <code>_StrictWeakOrdering</code> | Comparison function defining sort order. |
|----------------------------------|------------------------------------------|

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__x</code>    | Sorted list to merge. |
| <code>__comp</code> | Comparison functor.   |

Assumes that both `__x` and this list are sorted according to `StrictWeakOrdering`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equivalent according to `StrictWeakOrdering()`.

References [list\(\)](#), [std::\\_\\_addressof\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [end\(\)](#), and [merge\(\)](#).

**operator=()** [1/3]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc > & list::operator= (
    const list< _Tp, _Alloc > & __x)
```

List assignment operator.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

All the elements of `__x` are copied.

Whether the allocator is copied depends on the allocator traits.

References [list\(\)](#), [std::\\_\\_addressof\(\)](#), [begin\(\)](#), [clear\(\)](#), and [end\(\)](#).

**operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list & std::list< _Tp, _Alloc >::operator= (
    initializer_list< value_type > __l) [inline]
```

List initializer list assignment operator.

**Parameters**

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> of <code>value_type</code> . |
|------------------|---------------------------------------------------------------|

Replace the contents of the list with copies of the elements in the `initializer_list __l`. This is linear in `l.size()`.

**operator=()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list & std::list<_Tp, _Alloc>::operator= (
    list<_Tp, _Alloc> && __x) [inline], [noexcept]
```

List move assignment operator.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

The contents of `__x` are moved into this list (without copying).

Afterwards `__x` is a valid, but unspecified list

Whether the allocator is moved depends on the allocator traits.

**pop\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::pop_back () [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

**pop\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::pop_front () [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

**push\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::push_back (
    const value_type & __x) [inline]
```

Add data to the end of the list.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.



**push\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_front (
    const value_type & __x) [inline]
```

Add data to the front of the list.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

**rbegin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rbegin () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

**rbegin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rbegin () [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

**remove()**

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::__remove_return_type list::remove (
    const _Tp & __value)
```

Remove all elements equal to value.

**Parameters**

|                      |                      |
|----------------------|----------------------|
| <code>__value</code> | The value to remove. |
|----------------------|----------------------|

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [get\\_allocator\(\)](#), [size\(\)](#), and [splice\(\)](#).

**remove\_if()**

```
template<typename _Tp, typename _Alloc>
template<typename _Predicate>
list< _Tp, _Alloc >::__remove_return_type list::remove_if (
    _Predicate __pred)
```

Remove all elements satisfying a predicate.

## Template Parameters

|                         |                                     |
|-------------------------|-------------------------------------|
| <code>_Predicate</code> | Unary predicate function or object. |
|-------------------------|-------------------------------------|

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [get\\_allocator\(\)](#), [size\(\)](#), and [splice\(\)](#).

**rend()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list<_Tp, _Alloc>::rend () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

**rend()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list<_Tp, _Alloc>::rend () [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

**resize()** [1/2]

```
template<typename _Tp, typename _Alloc>
void list::resize (
    size_type __new_size)
```

Resizes the list to the specified number of elements.

## Parameters

|                         |                                             |
|-------------------------|---------------------------------------------|
| <code>__new_size</code> | Number of elements the list should contain. |
|-------------------------|---------------------------------------------|

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

References [std::end\(\)](#), and [erase\(\)](#).

**resize()** [2/2]

```
template<typename _Tp, typename _Alloc>
void list::resize (
    size_type __new_size,
    const value_type & __x)
```

Resizes the list to the specified number of elements.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the list should contain.       |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

References [std::end\(\)](#), [erase\(\)](#), and [insert\(\)](#).

**reverse()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::reverse () [inline], [noexcept]
```

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

**size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::size () const [inline], [nodiscard], [noexcept]
```

Returns the number of elements in the list.

Referenced by [std::operator<=>\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), [unique\(\)](#), and [unique\(\)](#).

**sort()** [1/2]

```
template<typename _Tp, typename _Alloc>
void list::sort ()
```

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

References [std::begin\(\)](#), [std::empty\(\)](#), and [std::end\(\)](#).

**sort()** [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _StrictWeakOrdering>
void list::sort (
```

```
    _StrictWeakOrdering __comp)
```

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

References [std::begin\(\)](#), [std::empty\(\)](#), and [std::end\(\)](#).

**splice()** [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
    list< _Tp, _Alloc > && __x) [inline], [noexcept]
```

Insert contents of another list.

**Parameters**

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |

The elements of `__x` are inserted in constant time in front of the element referenced by `__position`. `__x` becomes an empty list.

Requires this `!= __x`.

Referenced by [insert\(\)](#), [insert\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), [unique\(\)](#), and [unique\(\)](#).

**splice()** [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
```

```
list< _Tp, _Alloc > && __x,
const_iterator __first,
const_iterator __last) [inline], [noexcept]
```

Insert range from another list.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |
| <code>__first</code>    | Iterator referencing the start of range in x.      |
| <code>__last</code>     | Iterator referencing the end of range in x.        |

Removes elements in the range [`__first`,`__last`) and inserts them before `__position` in constant time. Undefined if `__position` is in [`__first`,`__last`).

#### splice() [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
    list< _Tp, _Alloc > && __x,
    const_iterator __i) [inline], [noexcept]
```

Insert element from another list.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |
| <code>__i</code>        | Iterator referencing the element to move.          |

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

#### splice() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
    list< _Tp, _Alloc > & __x,
    const_iterator __first,
    const_iterator __last) [inline], [noexcept]
```

Insert range from another list.

#### Parameters

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>__position</code> | Const_iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                             |
| <code>__first</code>    | Const_iterator referencing the start of range in x.      |
| <code>__last</code>     | Const_iterator referencing the end of range in x.        |

Removes elements in the range [`__first`,`__last`) and inserts them before `__position` in constant time. Undefined if `__position` is in [`__first`,`__last`).

**splice()** [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
    list< _Tp, _Alloc > & __x,
    const_iterator __i) [inline], [noexcept]
```

Insert element from another list.

**Parameters**

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |
| <code>__i</code>        | Iterator referencing the element to move.          |

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

**swap()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap (
    list< _Tp, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another list.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A list of the same element and allocator types. |
|------------------|-------------------------------------------------|

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

**unique()** [1/2]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::__remove_return_type list::unique ()
```

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [get\\_allocator\(\)](#), [size\(\)](#), and [splice\(\)](#).

**unique()** [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _BinaryPredicate>
list< _Tp, _Alloc >::__remove_return_type list::unique (
    _BinaryPredicate __binary_pred)
```

Remove consecutive elements satisfying a predicate.

**Template Parameters**

|                               |                                      |
|-------------------------------|--------------------------------------|
| <code>_BinaryPredicate</code> | Binary predicate function or object. |
|-------------------------------|--------------------------------------|

For each consecutive set of elements  $[first, last)$  that satisfy  $predicate(first, i)$  where  $i$  is an iterator in  $[first, last)$ , remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [get\\_allocator\(\)](#), [size\(\)](#), and [splice\(\)](#).

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

## 5.634 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >` Class Template Reference

```
#include <assoc_container.hpp>
```

### Public Types

- typedef [list\\_update\\_tag](#) **container\_category**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Update_Policy` **update\_policy**

### Public Member Functions

- **list\_update** (const [list\\_update](#) &other)
- template<typename `It`>  
[list\\_update](#) (`It` first, `It` last)
- [list\\_update](#) & **operator=** (const [list\\_update](#) &other)
- void **swap** ([list\\_update](#) &other)

#### 5.634.1 Detailed Description

```
template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type, class  
Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>  
class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >
```

A list-update based associative container.

#### Template Parameters

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                         |
| <i>Mapped</i>        | Map type.                                                                         |
| <i>Eq_Fn</i>         | Equal functor.                                                                    |
| <i>Update_Policy</i> | Update policy, determines when an element will be moved to the front of the list. |
| <i>_Alloc</i>        | Allocator type.                                                                   |

Base is `detail::lu_map`.

### 5.634.2 Constructor & Destructor Documentation

#### list\_update()

```
template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type,
class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update (
    It first,
    It last) [inline]
```

Constructor taking \_\_iterators to a range of value\_types. The value\_types between first\_it and last\_it will be inserted into the container object.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

### 5.635 \_\_gnu\_pbds::list\_update\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::list\_update\_tag:



#### 5.635.1 Detailed Description

List-update.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.636 std::locale Class Reference

```
#include <locale_classes.h>
```

#### Classes

- class [facet](#)
- class [id](#)

## Public Types

- typedef int [category](#)

## Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const char \*\_\_s)
- [locale](#) (const [locale](#) &\_\_base, const char \*\_\_s, [category](#) \_\_cat)
- [locale](#) (const [locale](#) &\_\_base, const [locale](#) &\_\_add, [category](#) \_\_cat)
- [locale](#) (const [locale](#) &\_\_base, const [std::string](#) &\_\_s, [category](#) \_\_cat)
- [locale](#) (const [locale](#) &\_\_other) throw ()
- template<typename \_Facet>  
[locale](#) (const [locale](#) &\_\_other, \_Facet \*\_\_f)
- [locale](#) (const [std::string](#) &\_\_s)
- [~locale](#) () throw ()
- template<typename \_Facet>  
[locale combine](#) (const [locale](#) &\_\_other) const
- [string name](#) () const
- template<typename \_Char, typename \_Traits, typename \_Alloc>  
bool [operator\(\)](#) (const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s2) const
- template<typename \_CharT, typename \_Traits, typename \_Alloc>  
bool [operator\(\)](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s2) const
- const [locale](#) & [operator=](#) (const [locale](#) &\_\_other) throw ()
- bool [operator==](#) (const [locale](#) &\_\_other) const throw ()

## Static Public Member Functions

- static const [locale](#) & [classic](#) ()
- static [locale global](#) (const [locale](#) &\_\_loc)

## Static Public Attributes

- static const [category](#) [none](#)
- static const [category](#) [ctype](#)
- static const [category](#) [numeric](#)
- static const [category](#) [collate](#)
- static const [category](#) [time](#)
- static const [category](#) [monetary](#)
- static const [category](#) [messages](#)
- static const [category](#) [all](#)

## Friends

- template<typename \_Facet>  
const \_Facet \* [\\_\\_try\\_use\\_facet](#) (const [locale](#) &\_\_loc)
- template<typename \_Cache>  
struct [\\_\\_use\\_cache](#)
- class [\\_Impl](#)
- class [facet](#)



- `template<typename _Facet>`  
`bool has_facet (const locale &__loc) throw ()`
- `template<typename _Facet>`  
`const _Facet & use_facet (const locale &__loc)`

### 5.636.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing. Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

### 5.636.2 Member Typedef Documentation

#### category

```
typedef int std::locale::category
```

Definition of locale::category.

### 5.636.3 Constructor & Destructor Documentation

#### locale() [1/8]

```
std::locale::locale () throw ( )
```

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [~locale\(\)](#), [combine\(\)](#), [global\(\)](#), [has\\_facet](#), [operator\(\)\(\)](#), [operator=\(\)](#), [operator==\(\)](#), and [use\\_facet](#).

#### locale() [2/8]

```
std::locale::locale (
    const locale & __other) throw ( )
```

Copy constructor.

Constructs a copy of *other*.

#### Parameters

|                      |                     |
|----------------------|---------------------|
| <code>__other</code> | The locale to copy. |
|----------------------|---------------------|

References [locale\(\)](#).

#### locale() [3/8]

```
std::locale::locale (
    const char * __s) [explicit]
```

Named locale constructor.

Constructs a copy of the named C library locale.

#### Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__s</code> | Name of the locale to construct. |
|------------------|----------------------------------|

## Exceptions

|                                 |                                                     |
|---------------------------------|-----------------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is null or an undefined locale. |
|---------------------------------|-----------------------------------------------------|

References [locale\(\)](#).

**locale()** [4/8]

```
std::locale::locale (
    const locale & __base,
    const char * __s,
    category __cat)
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

## Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                                  |
| <code>__s</code>    | Name of the locale to use facets from.                               |
| <code>__cat</code>  | Set of categories defining the facets to use from <code>__s</code> . |

## Exceptions

|                                 |                                                     |
|---------------------------------|-----------------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is null or an undefined locale. |
|---------------------------------|-----------------------------------------------------|

References [locale\(\)](#).

**locale()** [5/8]

```
std::locale::locale (
    const std::string & __s) [inline], [explicit]
```

Named locale constructor.

Constructs a copy of the named C library locale.

## Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__s</code> | Name of the locale to construct. |
|------------------|----------------------------------|

## Exceptions

|                                 |                                             |
|---------------------------------|---------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is an undefined locale. |
|---------------------------------|---------------------------------------------|

References [locale\(\)](#).

**locale()** [6/8]

```
std::locale::locale (
    const locale & __base,
```

```
const std::string & __s,
category __cat) [inline]
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

#### Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                                  |
| <code>__s</code>    | Name of the locale to use facets from.                               |
| <code>__cat</code>  | Set of categories defining the facets to use from <code>__s</code> . |

#### Exceptions

|                                 |                                             |
|---------------------------------|---------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is an undefined locale. |
|---------------------------------|---------------------------------------------|

References [locale\(\)](#).

#### locale() [7/8]

```
std::locale::locale (
    const locale & __base,
    const locale & __add,
    category __cat)
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

#### Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                            |
| <code>__add</code>  | The locale to use facets from.                                 |
| <code>__cat</code>  | Set of categories defining the facets to use from <i>add</i> . |

References [locale\(\)](#).

#### locale() [8/8]

```
template<typename _Facet>
std::locale::locale (
    const locale & __other,
    _Facet * __f)
```

Construct locale with another facet.

Constructs a copy of the locale `__other`. The facet `__f` is added to `__other`, replacing an existing facet of type `Facet` if there is one. If `__f` is null, this locale is a copy of `__other`.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__other</code> | The locale to copy.  |
| <code>__f</code>     | The facet to add in. |

References [locale\(\)](#).

**~locale()**

```
std::locale::~~locale () throw ( )
```

Locale destructor.

References [locale\(\)](#).

**5.636.4 Member Function Documentation****classic()**

```
static const locale & std::locale::classic () [static], [nodiscard]
```

Return reference to the C locale.

**combine()**

```
template<typename _Facet>
locale std::locale::combine (
    const locale & __other) const [nodiscard]
```

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type *Facet* from the locale *other* into the new locale.

**Template Parameters**

|                     |                                   |
|---------------------|-----------------------------------|
| <code>_Facet</code> | The facet type to copy from other |
|---------------------|-----------------------------------|

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__other</code> | The locale to copy from. |
|----------------------|--------------------------|

**Returns**

Newly constructed locale.

**Exceptions**

|                                 |                                                                    |
|---------------------------------|--------------------------------------------------------------------|
| <code>std::runtime_error</code> | if <code>__other</code> has no facet of type <code>_Facet</code> . |
|---------------------------------|--------------------------------------------------------------------|

References [locale\(\)](#).

Referenced by [operator=\(\)](#).

**global()**

```
static locale std::locale::global (
    const locale & __loc) [static]
```

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call `std::setlocale(LC_ALL, loc.name())`.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__loc</code> | The new locale to make global. |
|--------------------|--------------------------------|

**Returns**

Copy of the old global locale.

References [locale\(\)](#).

**name()**

```
string std::locale::name () const [nodiscard]
```

Return locale name.

**Returns**

Locale name or "\*" if unnamed.

References [name\(\)](#).

Referenced by [name\(\)](#).

**operator>()()**

```
template<typename _Char, typename _Traits, typename _Alloc>
bool std::locale::operator() (
    const basic_string< _Char, _Traits, _Alloc > & __s1,
    const basic_string< _Char, _Traits, _Alloc > & __s2) const [nodiscard]
```

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector *v* of strings could be sorted according to locale *loc* by doing:

```
std::sort(v.begin(), v.end(), loc);
```

**Parameters**

|                   |                           |
|-------------------|---------------------------|
| <code>__s1</code> | First string to compare.  |
| <code>__s2</code> | Second string to compare. |

**Returns**

True if `collate<_Char> facet` compares `__s1 < __s2`, else false.

References [locale\(\)](#).

**operator=()**

```
const locale & std::locale::operator= (
    const locale & __other) throw ( )
```

Assignment operator.

Set this locale to be a copy of *other*.

**Parameters**

|                      |                     |
|----------------------|---------------------|
| <code>__other</code> | The locale to copy. |
|----------------------|---------------------|

**Returns**

A reference to this locale.

References [locale\(\)](#), and [combine\(\)](#).

**operator==()**

```
bool std::locale::operator== (
    const locale & __other) const throw ( )    [nodiscard]
```

Locale equality.

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__other</code> | The locale to compare against. |
|----------------------|--------------------------------|

**Returns**

True if other and this refer to the same locale instance, are copies, or have the same name. False otherwise.

References [locale\(\)](#), and [operator==\(\)](#).

Referenced by [operator==\(\)](#).

**5.636.5 Friends And Related Symbol Documentation****has\_facet**

```
template<typename _Facet>
bool has_facet (
    const locale & __loc) throw ( )    [friend]
```

Test for the presence of a facet.

has\_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

**Template Parameters**

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>_Facet</code> | The facet type to test the presence of. |
|---------------------|-----------------------------------------|

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__loc</code> | The locale to test. |
|--------------------|---------------------|

**Returns**

true if \_\_loc contains a facet of type \_Facet, else false.

References [locale\(\)](#).

**use\_facet**

```
template<typename _Facet>
const _Facet & use_facet (
    const locale & __loc)    [friend]
```

Return a facet.

use\_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type Facet.

**Template Parameters**

|                     |                           |
|---------------------|---------------------------|
| <code>_Facet</code> | The facet type to access. |
|---------------------|---------------------------|

#### Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__loc</code> | The locale to use. |
|--------------------|--------------------|

#### Returns

Reference to facet of type `Facet`.

#### Exceptions

|                            |                                                                             |
|----------------------------|-----------------------------------------------------------------------------|
| <code>std::bad_cast</code> | if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> . |
|----------------------------|-----------------------------------------------------------------------------|

References [locale\(\)](#).

### 5.636.6 Member Data Documentation

#### **all**

```
const category std::locale::all [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

#### **collate**

```
const category std::locale::collate [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

#### **ctype**

```
const category std::locale::ctype [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

#### **messages**

```
const category std::locale::messages [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

#### **monetary**

```
const category std::locale::monetary [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

**none**

```
const category std::locale::none [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

**numeric**

```
const category std::locale::numeric [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

**time**

```
const category std::locale::time [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

**5.637 std::lock\_guard< \_Mutex > Class Template Reference**

```
#include <mutex>
```

**Public Types**

- typedef `_Mutex` **mutex\_type**

**Public Member Functions**

- **lock\_guard** (const [lock\\_guard](#) &)=delete
- **lock\_guard** (mutex\_type &\_\_m)
- **lock\_guard** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#)) noexcept
- **lock\_guard** & **operator=** (const [lock\\_guard](#) &)=delete

**5.637.1 Detailed Description**

```
template<typename _Mutex>
```

```
class std::lock_guard< _Mutex >
```

A simple scoped lock type.

A `lock_guard` controls mutex ownership within a scope, releasing ownership in the destructor.

Since

C++11

The documentation for this class was generated from the following file:

- [std\\_mutex.h](#)



## 5.638 std::logic\_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for std::logic\_error:



### Public Member Functions

- **logic\_error** (const char \*)
- **logic\_error** (const [logic\\_error](#) &) noexcept
- **logic\_error** (const [string](#) & \_\_arg)
- **logic\_error** ([logic\\_error](#) &&) noexcept
- **logic\_error** & **operator=** (const [logic\\_error](#) &) noexcept
- **logic\_error** & **operator=** ([logic\\_error](#) &&) noexcept
- virtual const char \* **what** () const noexcept

### 5.638.1 Detailed Description

One of two subclasses of exception.

Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

### 5.638.2 Constructor & Destructor Documentation

#### logic\_error()

```
std::logic_error::logic_error (
    const string & __arg) [explicit]
```

Takes a character string describing the error.

### 5.638.3 Member Function Documentation

#### what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.639 std::logical\_and<\_Tp> Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical\_and<\_Tp>:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) const

### 5.639.1 Detailed Description

```
template<typename _Tp>
struct std::logical_and<_Tp>
```

One of the [Boolean operations functors](#).

### 5.639.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

**result\_type**

```
typedef bool std::binary\_function< _Tp, _Tp, bool >::result_type [inherited]
```

result\_type is the return type

**second\_argument\_type**

```
typedef _Tp std::binary\_function< _Tp, _Tp, bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

**5.640 std::logical\_and< void > Struct Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::logical\_and< void >:

**Public Types**

- typedef void [first\\_argument\\_type](#)
- typedef \_\_is\_transparent [is\\_transparent](#)
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

**Public Member Functions**

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t) &&[std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t) &&[std::forward](#)< \_Up >(\_\_u))
- constexpr bool **operator()** (const void &\_\_x, const void &\_\_y) const

**5.640.1 Detailed Description**

One of the [Boolean operations functors](#).

### 5.640.2 Member Typedef Documentation

#### first\_argument\_type

typedef void [std::binary\\_function](#)< void, void, bool >::first\_argument\_type

first\_argument\_type is the type of the first argument

#### result\_type

typedef bool [std::binary\\_function](#)< void, void, bool >::result\_type

result\_type is the return type

#### second\_argument\_type

typedef void [std::binary\\_function](#)< void, void, bool >::second\_argument\_type

second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.641 std::logical\_not<\_Tp> Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical\_not<\_Tp>:



### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef bool [result\\_type](#)

### Public Member Functions

- constexpr bool **operator()** (const `_Tp` &`__x`) const

### 5.641.1 Detailed Description

**template**<typename `_Tp`>

**struct** `std::logical_not<_Tp>`

One of the [Boolean operations functors](#).

### 5.641.2 Member Typedef Documentation

#### argument\_type

typedef \_Tp [std::unary\\_function](#)< \_Tp, bool >::argument\_type [inherited]  
 argument\_type is the type of the argument

#### result\_type

typedef bool [std::unary\\_function](#)< \_Tp, bool >::result\_type [inherited]  
 result\_type is the return type

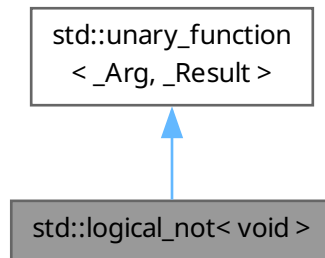
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.642 std::logical\_not< void > Struct Reference

#include <stl\_function.h>

Inheritance diagram for std::logical\_not< void >:



#### Public Types

- typedef void [argument\\_type](#)
- typedef \_\_is\_transparent [is\\_transparent](#)
- typedef bool [result\\_type](#)

#### Public Member Functions

- template<typename \_Tp>  
 constexpr auto **operator()** (\_Tp &&\_\_t) const noexcept(noexcept(![std::forward](#)< \_Tp >(\_\_t))) -> decltype(![std::forward](#)< \_Tp >(\_\_t))
- constexpr bool **operator()** (const void &\_\_x) const

#### 5.642.1 Detailed Description

One of the [Boolean operations functors](#).

### 5.642.2 Member Typedef Documentation

#### argument\_type

typedef void [std::unary\\_function](#)< void, bool >::argument\_type

argument\_type is the type of the argument

#### result\_type

typedef bool [std::unary\\_function](#)< void, bool >::result\_type

result\_type is the return type

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.643 std::logical\_or< \_Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical\_or< \_Tp >:



### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

### Public Member Functions

- constexpr bool **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

### 5.643.1 Detailed Description

template<typename \_Tp>

struct std::logical\_or< \_Tp >

One of the [Boolean operations functors](#).

### 5.643.2 Member Typedef Documentation

#### first\_argument\_type

typedef `_Tp` `std::binary_function`< `_Tp`, `_Tp`, `bool` >::`first_argument_type` [inherited]  
`first_argument_type` is the type of the first argument

#### result\_type

typedef `bool` `std::binary_function`< `_Tp`, `_Tp`, `bool` >::`result_type` [inherited]  
`result_type` is the return type

#### second\_argument\_type

typedef `_Tp` `std::binary_function`< `_Tp`, `_Tp`, `bool` >::`second_argument_type` [inherited]  
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.644 `std::logical_or`< `void` > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::logical_or`< `void` >:



#### Public Types

- typedef `void` [first\\_argument\\_type](#)
- typedef `__is_transparent` `is_transparent`
- typedef `bool` [result\\_type](#)
- typedef `void` [second\\_argument\\_type](#)

#### Public Member Functions

- template<typename `_Tp`, typename `_Up`>  
 constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`< `_Tp` >(`_t`) || `std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`) || `std::forward`< `_Up` >(`_u`))
- constexpr `bool` **operator()** (const `void` &`_x`, const `void` &`_y`) const

### 5.644.1 Detailed Description

One of the [Boolean operations functors](#).

### 5.644.2 Member Typedef Documentation

#### `first_argument_type`

typedef void `std::binary_function< void, void, bool >::first_argument_type`  
`first_argument_type` is the type of the first argument

#### `result_type`

typedef bool `std::binary_function< void, void, bool >::result_type`  
`result_type` is the return type

#### `second_argument_type`

typedef void `std::binary_function< void, void, bool >::second_argument_type`  
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.645 `std::lognormal_distribution<_RealType>` Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `lognormal_distribution` (`_RealType` \_\_m, `_RealType` \_\_s=`_RealType`(1))
- `lognormal_distribution` (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator`>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- `_RealType` m () const
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) operator() (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) operator() (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)



- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`
- `_RealType s () const`

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::lognormal_distribution< _RealType1 > &__x)`
- `bool operator== (const lognormal_distribution &__d1, const lognormal_distribution &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::lognormal_distribution< _RealType1 > &__x)`

### 5.645.1 Detailed Description

`template<typename _RealType = double>  
class std::lognormal_distribution< _RealType >`

A lognormal\_distribution random number distribution.  
The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Since

C++11

### 5.645.2 Member Typedef Documentation

#### result\_type

`template<typename _RealType = double>  
typedef _RealType std::lognormal_distribution< _RealType >::result_type`  
The type of the range of the distribution.

### 5.645.3 Member Function Documentation

#### max()

`template<typename _RealType = double>  
result_type std::lognormal_distribution< _RealType >::max () const [inline]`  
Returns the least upper bound value of the distribution.

#### min()

`template<typename _RealType = double>  
result_type std::lognormal_distribution< _RealType >::min () const [inline]`  
Returns the greatest lower bound value of the distribution.

**operator()()**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::lognormal_distribution<_RealType>::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by `operator()()`.

**param() [1/2]**

```
template<typename _RealType = double>
param_type std::lognormal_distribution<_RealType>::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _RealType = double>
void std::lognormal_distribution<_RealType>::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::lognormal_distribution<_RealType>::reset () [inline]
```

Resets the distribution state.

**5.645.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::lognormal_distribution<_RealType1> & __x) [friend]
```

Inserts a `lognormal_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                 |
| <code>__x</code>  | A <code>lognormal_distribution</code> random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
    const lognormal_distribution< _RealType > & __d1,
    const lognormal_distribution< _RealType > & __d2) [friend]
```

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::lognormal_distribution< _RealType1 > & __x) [friend]
```

Extracts a lognormal\_distribution random number distribution \_\_x from the input stream \_\_is.

**Parameters**

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| <code>__is</code> | An input stream.                                         |
| <code>__x</code>  | A lognormal_distribution random number generator engine. |

**Returns**

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.646 \_\_gnu\_pbds::detail::lu\_counter\_metadata< Size\_Type > Class Template Reference**

```
#include <lu_counter_metadata.hpp>
```

**Public Types**

- typedef Size\_Type **size\_type**

**Friends**

- class **lu\_counter\_policy\_base< size\_type >**

**5.646.1 Detailed Description**

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::detail::lu_counter_metadata< Size_Type >
```

A list-update metadata type that moves elements to the front of the list based on the counter algorithm.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

**5.647 `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >` Class Template Reference**

```
#include <list_update_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >`:

**Public Types**

- enum { `max_count` }
- typedef `_Alloc allocator_type`
- typedef `detail::rebind_traits< _Alloc, metadata_type >::reference metadata_reference`
- typedef `detail::lu_counter_metadata< size_type > metadata_type`
- typedef `allocator_type::size_type size_type`

**Public Member Functions**

- `metadata_type operator() () const`
- `bool operator() (metadata_reference r_data) const`

**Private Member Functions**

- `template<typename Metadata_Reference> bool operator() (Metadata_Reference r_data, size_type m_max_count) const`
- `lu_counter_metadata< size_type > operator() (size_type max_size) const`

**5.647.1 Detailed Description**

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >
```

A list-update policy that moves elements to the front of the list based on the counter algorithm.

**5.647.2 Member Typedef Documentation****metadata\_reference**

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc, metadata_type>::reference __gnu_pbds::lu_counter_policy<
Max_Count, _Alloc >::metadata_reference
```

Reference to metadata on which this functor operates.

#### metadata\_type

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::lu\_counter\_metadata<size_type> \_\_gnu\_pbds::lu\_counter\_policy< Max_Count, _Alloc
>::metadata_type
```

Metadata on which this functor operates.

### 5.647.3 Member Enumeration Documentation

#### anonymous enum

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
anonymous enum
```

#### Enumerator

|           |                                                                                                |
|-----------|------------------------------------------------------------------------------------------------|
| max_count | When some element is accessed this number of times, it will be moved to the front of the list. |
|-----------|------------------------------------------------------------------------------------------------|

### 5.647.4 Member Function Documentation

#### operator>() [1/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
metadata\_type \_\_gnu\_pbds::lu\_counter\_policy< Max_Count, _Alloc >::operator() () const [inline]
```

Creates a metadata object.

References [max\\_count](#).

#### operator>() [2/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
bool \_\_gnu\_pbds::lu\_counter\_policy< Max_Count, _Alloc >::operator() (
    metadata\_reference r_data) const [inline]
```

Decides whether a metadata object should be moved to the front of the list.

References [max\\_count](#).

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

### 5.648 [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base](#)< Size\_Type > Class Template Reference

```
#include <lu_counter_metadata.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::lu_counter_policy_base< Size_Type >`:



#### Protected Types

- typedef `Size_Type` **size\_type**

#### Protected Member Functions

- template<typename Metadata\_Reference>  
bool **operator()** (Metadata\_Reference r\_data, size\_type m\_max\_count) const
- [lu\\_counter\\_metadata](#)< size\_type > **operator()** (size\_type max\_size) const

##### 5.648.1 Detailed Description

```
template<typename Size_Type>  
class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >
```

Base class for list-update counter policy.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

## 5.649 `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` Class Template Reference

```
#include <lu_map_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**

- typedef `point_const_iterator_` **point\_const\_iterator**
- typedef `point_iterator_` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `Update_Policy::metadata_type` **update\_metadata**
- typedef `Update_Policy` **update\_policy**
- typedef `traits_base::value_type` **value\_type**

### Public Member Functions

- **lu\_map** (const `lu_map`< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy > &)
- template<typename It>  
  **lu\_map** (It, It)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred>  
  size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference r\_key)
- point\_const\_iterator **find** (key\_const\_reference r\_key) const
- `std::pair`< point\_iterator, bool > **insert** (const\_reference)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** (`lu_map`< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy > &)

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Member Functions

- template<typename It>  
  void **copy\_from\_range** (It, It)

### Friends

- class **const\_iterator\_**
- class **iterator\_**



### 5.649.1 Detailed Description

```
template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >
```

list-based (with updates) associative container. Skip to the lu, my darling.  
The documentation for this class was generated from the following file:

- [lu\\_map\\_.hpp](#)

## 5.650 \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc > Class Template Reference

```
#include <list_update_policy.hpp>
```

### Public Types

- typedef `_Alloc` `allocator_type`
- typedef `detail::rebind_traits< _Alloc, metadata_type >::reference` `metadata_reference`
- typedef `null_type` `metadata_type`

### Public Member Functions

- `metadata_type operator() () const`
- `bool operator() (metadata_reference r_metadata) const`

### 5.650.1 Detailed Description

```
template<typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_move_to_front_policy< _Alloc >
```

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

### 5.650.2 Member Typedef Documentation

#### `metadata_reference`

```
template<typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc,metadata_type>::reference __gnu_pbds::lu_move_to_front_policy<
_Alloc >::metadata_reference
```

Reference to metadata on which this functor operates.

#### `metadata_type`

```
template<typename _Alloc = std::allocator<char>>
typedef null_type __gnu_pbds::lu_move_to_front_policy< _Alloc >::metadata_type
```

Metadata on which this functor operates.

### 5.650.3 Member Function Documentation

#### `operator()()` [1/2]

```
template<typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_move_to_front_policy< _Alloc >::operator() () const [inline]
```

Creates a metadata object.

**operator()()** [2/2]

```
template<typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_move_to_front_policy<_Alloc>::operator() (
    metadata_reference r_metadata) const [inline]
```

Decides whether a metadata object should be moved to the front of the list.

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

**5.651 `std::make_signed<_Tp>` Struct Template Reference**

```
#include <type_traits>
```

**Public Types**

- using **type**

**5.651.1 Detailed Description**

```
template<typename _Tp>
struct std::make_signed<_Tp>
```

`make_signed`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.652 `std::make_unsigned<_Tp>` Struct Template Reference**

```
#include <type_traits>
```

**Public Types**

- using **type**

**5.652.1 Detailed Description**

```
template<typename _Tp>
struct std::make_unsigned<_Tp>
```

`make_unsigned`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.653 `__gnu_cxx::malloc_allocator<_Tp>` Class Template Reference**

```
#include <malloc_allocator.h>
```

**Public Types**

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **malloc\_allocator** (const [malloc\\_allocator](#) &) noexcept
- template<typename [\\_Tp1](#)>  
constexpr **malloc\_allocator** (const [malloc\\_allocator](#)< [\\_Tp1](#) > &) noexcept
- [\\_Tp](#) \* **allocate** (size\_type \_\_n, const void \*=0)
- void **deallocate** ([\\_Tp](#) \*\_\_p, size\_type)

## Friends

- template<typename [\\_Up](#)>  
constexpr bool **operator==** (const [malloc\\_allocator](#) &, const [malloc\\_allocator](#)< [\\_Up](#) > &) noexcept

### 5.653.1 Detailed Description

template<typename [\\_Tp](#)>  
class [\\_\\_gnu\\_cxx::malloc\\_allocator](#)< [\\_Tp](#) >

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free

The documentation for this class was generated from the following files:

- [experimental/memory\\_resource](#)
- [malloc\\_allocator.h](#)

## 5.654 std::\_\_debug::map< [\\_Key](#), [\\_Tp](#), [\\_Compare](#), [\\_Allocator](#) > Class Template Reference

#include <map.h>

Inheritance diagram for std::\_\_debug::map< [\\_Key](#), [\\_Tp](#), [\\_Compare](#), [\\_Allocator](#) >:



## Public Types

- typedef [\\_Allocator](#) **allocator\_type**
- typedef [\\_\\_gnu\\_debug::Safe\\_iterator](#)< [\\_Base\\_const\\_iterator](#), [map](#) > **const\_iterator**
- typedef [\\_Base::const\\_pointer](#) **const\_pointer**
- typedef [\\_Base::const\\_reference](#) **const\_reference**
- typedef [std::reverse\\_iterator](#)< [const\\_iterator](#) > **const\_reverse\_iterator**
- typedef [\\_Base::difference\\_type](#) **difference\_type**
- using **insert\_return\_type**

- typedef `__gnu_debug::Safe_iterator<_Base_iterator, map>` `iterator`
- typedef `_Compare` `key_compare`
- typedef `_Key` `key_type`
- typedef `_Tp` `mapped_type`
- using `node_type`
- typedef `_Base::pointer` `pointer`
- typedef `_Base::reference` `reference`
- typedef `std::reverse_iterator<iterator>` `reverse_iterator`
- typedef `_Base::size_type` `size_type`
- typedef `std::pair<const _Key, _Tp>` `value_type`

## Public Member Functions

- `map` (`_Base_ref __x`)
- template<typename `_InputIterator`>  
`map` (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- template<typename `_InputIterator`>  
`map` (`_InputIterator __first`, `_InputIterator __last`, `const allocator_type &__a`)
- `map` (`const _Compare &__comp`, `const _Allocator &__a=_Allocator()`)
- `map` (`const allocator_type &__a`)
- `map` (`const map &`)=default
- `map` (`const map &__m`, `const __type_identity_t<allocator_type> &__a`)
- `map` (`initializer_list<value_type> __l`, `const _Compare &__c=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `map` (`initializer_list<value_type> __l`, `const allocator_type &__a`)
- `map` (`map &&`)=default
- `map` (`map &&__m`, `const __type_identity_t<allocator_type> &__a`) noexcept(noexcept(`_Base::move(__m, __a)`))
- `__attribute__((abi_tag("cxx11")))` `iterator` `erase(iterator __position)`
- `const _Base &_M_base` () const noexcept
- `_Base &_M_base` () noexcept
- `const_iterator` `begin` () const noexcept
- `iterator` `begin` () noexcept
- `const_iterator` `cbegin` () const noexcept
- `const_iterator` `cend` () const noexcept
- void `clear` () noexcept
- `const_reverse_iterator` `crbegin` () const noexcept
- `const_reverse_iterator` `crend` () const noexcept
- template<typename... `_Args`>  
`std::pair<iterator, bool>` `emplace` (`_Args &&... __args`)
- template<typename... `_Args`>  
`iterator` `emplace_hint` (`const_iterator __pos`, `_Args &&... __args`)
- `const_iterator` `end` () const noexcept
- `iterator` `end` () noexcept
- template<typename `_Kt`, typename `_Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair<iterator, iterator>` `equal_range` (`const _Kt &__x`)
- template<typename `_Kt`, typename `_Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair<const_iterator, const_iterator>` `equal_range` (`const _Kt &__x`) const
- `std::pair<iterator, iterator>` `equal_range` (`const key_type &__x`)
- `std::pair<const_iterator, const_iterator>` `equal_range` (`const key_type &__x`) const
- `_Base_iterator` `erase` (`_Base_const_iterator __position`)

- `size_type erase (const key_type &__x)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator erase (const_iterator __position)`
- `node_type extract (const key_type &__key)`
- `node_type extract (const_iterator __position)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> iterator find (const _Kt &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> const_iterator find (const _Kt &__x) const`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `template<typename _InputIterator> void insert (_InputIterator __first, _InputIterator __last)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> std::pair< iterator, bool > insert (_Pair &&__x)`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `iterator insert (const_iterator __hint, node_type &&__nh)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator insert (const_iterator __position, _Pair &&__x)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type &&__x)`
- `insert_return_type insert (node_type &&__nh)`
- `void insert (std::initializer_list< value_type > __list)`
- `std::pair< iterator, bool > insert (value_type &&__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> iterator lower_bound (const _Kt &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> const_iterator lower_bound (const _Kt &__x) const`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `map & operator= (const map &)=default`
- `map & operator= (initializer_list< value_type > __l)`
- `map & operator= (map &&)=default`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `reverse_iterator rend () noexcept`
- `void swap (map &__x) noexcept(*conditional *)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> iterator upper_bound (const _Kt &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> const_iterator upper_bound (const _Kt &__x) const`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`

## Protected Member Functions

- `constexpr void _M_swap (const _Safe_container &__x) const noexcept`

## Friends

- `template<typename _ItT, typename _SeqT, typename _CatT> class ::__gnu_debug:: _Safe_iterator`

### 5.654.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator =
std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::map< _Key, _Tp, _Compare, _Allocator >
```

Class std::map with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [map.h](#)

## 5.655 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

```
#include <map>
```

### Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_Rep\_type::const\_iterator **const\_iterator**
- typedef \_Alloc\_traits::const\_pointer **const\_pointer**
- typedef \_Alloc\_traits::const\_reference **const\_reference**
- typedef \_Rep\_type::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Alloc\_traits::pointer **pointer**
- typedef \_Alloc\_traits::reference **reference**
- typedef \_Rep\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Rep\_type::size\_type **size\_type**
- typedef [std::pair](#)< const \_Key, \_Tp > **value\_type**

### Public Member Functions

- [map](#) ()=default
- template<typename \_InputIterator>  
[map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator>  
[map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator>  
[map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- [map](#) (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- [map](#) (const allocator\_type &\_\_a)
- [map](#) (const [map](#) &)=default
- [map](#) (const [map](#) &\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a)
- [map](#) (initializer\_list< [value\\_type](#) > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- [map](#) (initializer\_list< [value\\_type](#) > \_\_l, const allocator\_type &\_\_a)
- [map](#) ([map](#) &&)=default
- [map](#) ([map](#) &&\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept([is\\_nothrow\\_copy\\_constructible](#)< \_Compare >::value && \_Alloc\_traits::\_S\_always\_equal())
- [~map](#) ()=default

- mapped\_type & **at** (const key\_type &\_\_k)
  - const mapped\_type & **at** (const key\_type &\_\_k) const
  - const\_iterator **begin** () const noexcept
  - iterator **begin** () noexcept
  - const\_iterator **cbegin** () const noexcept
  - const\_iterator **end** () const noexcept
  - void **clear** () noexcept
  - const\_reverse\_iterator **crbegin** () const noexcept
  - const\_reverse\_iterator **crend** () const noexcept
  - template<typename... \_Args>  
std::pair< iterator, bool > **emplace** (\_Args &&... \_\_args)
  - template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool **empty** () const noexcept
  - const\_iterator **end** () const noexcept
  - iterator **end** () noexcept
  - size\_type **erase** (const key\_type &\_\_x)
  - iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
  - allocator\_type **get\_allocator** () const noexcept
  - template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void **insert** (std::initializer\_list< value\_type > \_\_list)
  - template<typename \_Obj>  
pair< iterator, bool > **insert\_or\_assign** (const key\_type &\_\_k, \_Obj &&\_\_obj)
  - template<typename \_Obj>  
iterator **insert\_or\_assign** (const\_iterator \_\_hint, const key\_type &\_\_k, \_Obj &&\_\_obj)
  - template<typename \_Obj>  
iterator **insert\_or\_assign** (const\_iterator \_\_hint, key\_type &&\_\_k, \_Obj &&\_\_obj)
  - template<typename \_Obj>  
pair< iterator, bool > **insert\_or\_assign** (key\_type &&\_\_k, \_Obj &&\_\_obj)
  - key\_compare **key\_comp** () const
  - size\_type **max\_size** () const noexcept
  - map & **operator=** (const map &)=default
  - map & **operator=** (initializer\_list< value\_type > \_\_l)
  - map & **operator=** (map &&)=default
  - mapped\_type & **operator[]** (const key\_type &\_\_k)
  - mapped\_type & **operator[]** (key\_type &&\_\_k)
  - const\_reverse\_iterator **rbegin** () const noexcept
  - reverse\_iterator **rbegin** () noexcept
  - const\_reverse\_iterator **rend** () const noexcept
  - reverse\_iterator **rend** () noexcept
  - size\_type **size** () const noexcept
  - void **swap** (map &\_\_x) noexcept(*/\*conditional \*/*)
  - value\_compare **value\_comp** () const
- 
- std::pair< iterator, bool > **insert** (const value\_type &\_\_x)
  - std::pair< iterator, bool > **insert** (value\_type &&\_\_x)
  - template<typename \_Pair>  
\_\_enable\_if\_t< is\_constructible< value\_type, \_Pair >::value, pair< iterator, bool > > **insert** (\_Pair &&\_\_x)

- iterator [insert](#) (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator [insert](#) (const\_iterator \_\_position, value\_type &&\_\_x)
- template<typename \_Pair>  
\_\_enable\_if\_t< is\_constructible< value\_type, \_Pair >::value, iterator > [insert](#) (const\_iterator \_\_position, \_Pair &&\_\_x)
- iterator [erase](#) (const\_iterator \_\_position)
- [\\_\\_attribute](#) ((\_\_abi\_tag\_\_("cxx11"))) iterator [erase](#)(iterator \_\_position)
- iterator [find](#) (const key\_type &\_\_x)
- template<typename \_Kt>  
auto [find](#) (const \_Kt &\_\_x) -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))
- const\_iterator [find](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [find](#) (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))
- size\_type [count](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [count](#) (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_count\_tr(\_\_x))
- bool [contains](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [contains](#) (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x), void(), true)
- iterator [lower\\_bound](#) (const key\_type &\_\_x)
- template<typename \_Kt>  
auto [lower\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- const\_iterator [lower\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [lower\\_bound](#) (const \_Kt &\_\_x) const -> decltype(const\_iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- iterator [upper\\_bound](#) (const key\_type &\_\_x)
- template<typename \_Kt>  
auto [upper\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
- const\_iterator [upper\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [upper\\_bound](#) (const \_Kt &\_\_x) const -> decltype(const\_iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))



- `std::pair< iterator, iterator > equal_range` (const key\_type &\_\_x)
- `template<typename _Kt>`  
`auto equal_range` (const \_Kt &\_\_x) -> decltype(pair< iterator, iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))
- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_x) const
- `template<typename _Kt>`  
`auto equal_range` (const \_Kt &\_\_x) const -> decltype(pair< const\_iterator, const\_iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))

## Friends

- `template<typename _K1, typename _T1, typename _C1, typename _A1>`  
`__detail::__synth3way_t< pair< const _K1, _T1 > > operator<=>` (const map< \_K1, \_T1, \_C1, \_A1 > &, const map< \_K1, \_T1, \_C1, \_A1 > &)
- `template<typename _K1, typename _T1, typename _C1, typename _A1>`  
`bool operator==` (const map< \_K1, \_T1, \_C1, \_A1 > &, const map< \_K1, \_T1, \_C1, \_A1 > &)

### 5.655.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >>`  
`class std::map< _Key, _Tp, _Compare, _Alloc >`

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Since

C++98

#### Template Parameters

|                       |                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                                |
| <code>_Tp</code>      | Type of mapped objects.                                                             |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .        |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (\*\_unique versus \*\_equal, same as the standard).

### 5.655.2 Constructor & Destructor Documentation

#### `map()` [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map () [default]
```

Default constructor creates no elements.

**map()** [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
    const _Compare & __comp,
    const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a map with no elements.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

**map()** [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
    const map<_Key, _Tp, _Compare, _Alloc > & ) [default]
```

Map copy constructor.

Whether the allocator is copied depends on the allocator traits.

**map()** [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
    map<_Key, _Tp, _Compare, _Alloc > && ) [default]
```

Map move constructor.

The newly-created map contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, map.

**map()** [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
    initializer_list<value_type> & __l,
    const _Compare & __comp = _Compare(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a map from an initializer\_list.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__l</code>    | An initializer_list. |
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

Create a map consisting of copies of the elements in the initializer\_list `__l`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `__l.size()`).

**map()** [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

**map()** [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    const map< _Key, _Tp, _Compare, _Alloc > & __m,
    const __type_identity_t< allocator_type > & __a) [inline]
```

Allocator-extended copy constructor.

**map()** [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    map< _Key, _Tp, _Compare, _Alloc > && __m,
    const __type_identity_t< allocator_type > & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

**map()** [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    initializer_list< value_type > __l,
    const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

**map()** [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

**map()** [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Builds a map from a range.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

**map()** [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
    _InputIterator __first,
    _InputIterator __last,
    const _Compare & __comp,
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a map from a range.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

**~map()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map<_Key, _Tp, _Compare, _Alloc >::~~map () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**5.655.3 Member Function Documentation****\_\_attribute()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map<_Key, _Tp, _Compare, _Alloc >::__attribute (
    (__abi_tag__("cxx11")) ) [inline]
```

Erases an element from a map.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### at()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type & std::map<_Key, _Tp, _Compare, _Alloc >::at (
    const key_type & __k) [inline]
```

Access to map data.

### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

### Returns

A reference to the data whose key is equivalent to `__k`, if such a data is present in the map.

### Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

### begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

### begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

### cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

**cend()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

**clear()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
void std::map<_Key, _Tp, _Compare, _Alloc>::clear () [inline], [noexcept]
```

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
template<typename _Kt>
```

```
auto std::map<_Key, _Tp, _Compare, _Alloc>::contains (
    const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                                                                                     |                                          |
|-------------------------------------------------------------------------------------|------------------------------------------|
|  | Key of (key, value) pairs to be located. |
| <b>__x</b>                                                                          |                                          |

**Returns**

True if there is an element with the specified key.

**contains()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
bool std::map<_Key, _Tp, _Compare, _Alloc>::contains (
    const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                                                                                     |                                          |
|-------------------------------------------------------------------------------------|------------------------------------------|
|  | Key of (key, value) pairs to be located. |
| <b>__x</b>                                                                          |                                          |

**Returns**

True if there is an element with the specified key.

**count()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::count (
    const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x))    [inline]
```

Finds the number of elements with given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

**count()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::count (
    const key_type & __x) const    [inline]
```

Finds the number of elements with given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

**crbegin()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crbegin () const    [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

**crend()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crend () const    [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

**emplace()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
std::pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::emplace (
    _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

**Parameters**

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

**emplace\_hint()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

**Parameters**

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                |
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the element with key of the std::pair built from \_\_args (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Referenced by `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace()`, `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert()`, `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign()`, and `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign()`.



**empty()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::map<_Key, _Tp, _Compare, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the map is empty. (Thus `begin()` would equal `end()`.)

**end() [1/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

**end() [2/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Referenced by `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::at()`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert()`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::operator[]()`.

**equal\_range() [1/4]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map<_Key, _Tp, _Compare, _Alloc >::equal_range (
    const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

**equal\_range() [2/4]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
```

```
auto std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
    const _Kt & __x) const -> decltype(pair<const_iterator, const_iterator> ( _M_t._M_↵
equal_range_tr(__x))) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                         |                                          |
|-------------------------|------------------------------------------|
| <code>_↵<br/>__X</code> | Key of (key, value) pairs to be located. |
|-------------------------|------------------------------------------|

#### Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

#### equal\_range() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, iterator > std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
    const key_type & __x) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                         |                                          |
|-------------------------|------------------------------------------|
| <code>_↵<br/>__X</code> | Key of (key, value) pairs to be located. |
|-------------------------|------------------------------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

#### equal\_range() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< const_iterator, const_iterator > std::map< _Key, _Tp, _Compare, _Alloc >::equal_range
(
    const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                         |                                          |
|-------------------------|------------------------------------------|
| <code>_↵<br/>__X</code> | Key of (key, value) pairs to be located. |
|-------------------------|------------------------------------------|

**Returns**

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

**erase()** [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map<_Key, _Tp, _Compare, _Alloc >::erase (
    const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a map.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::erase (
    const_iterator __position) [inline]
```

Erases an element from a map.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map<_Key, _Tp, _Compare, _Alloc >::find (
    const _Kt & __x) -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a map.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**find()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map<_Key, _Tp, _Compare, _Alloc >::find (
    const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a map.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::find (
    const key_type & __x) [inline]
```

Tries to locate an element in a map.

**Parameters**

|                                                                                      |                                         |
|--------------------------------------------------------------------------------------|-----------------------------------------|
|  _x | Key of (key, value) pair to be located. |
|--------------------------------------------------------------------------------------|-----------------------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::find (
    const key_type & __x) const [inline]
```

Tries to locate an element in a map.

**Parameters**

|                                                                                        |                                         |
|----------------------------------------------------------------------------------------|-----------------------------------------|
|  _x | Key of (key, value) pair to be located. |
|----------------------------------------------------------------------------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( end() ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::map<_Key, _Tp, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]
```

Get a copy of the memory allocation object.

**insert()** [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
void std::map<_Key, _Tp, _Compare, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert()** [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, pair< iterator, bool > > std::map<
_Key, _Tp, _Compare, _Alloc >::insert (
    _Pair && __x) [inline]
```

Attempts to insert a std::pair into the map.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map. Insertion requires logarithmic time.

**insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, bool > std::map<_Key, _Tp, _Compare, _Alloc >::insert (
    const value_type & __x) [inline]
```

Attempts to insert a std::pair into the map.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Referenced by `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert()`, and `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::operator[]()`.

**insert()** [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, iterator > std::map< _Key, _Tp, _C↵
_Compare, _Alloc >::insert (
    const_iterator __position,
    _Pair && __x) [inline]
```

Attempts to insert a std::pair into the map.

**Parameters**

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert (
    const_iterator __position,
    const value_type & __x) [inline]
```

Attempts to insert a std::pair into the map.

**Parameters**

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::map<_Key, _Tp, _Compare, _Alloc>::insert (
    const_iterator __position,
    value_type && __x) [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
void std::map<_Key, _Tp, _Compare, _Alloc>::insert (
    std::initializer_list<value_type> __list) [inline]
```

Attempts to insert a list of `std::pairs` into the map.

**Parameters**

|                     |                                                                                 |
|---------------------|---------------------------------------------------------------------------------|
| <code>__list</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of pairs to be inserted. |
|---------------------|---------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

**insert()** [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc>::insert (
    value_type && __x) [inline]
```

Attempts to insert a `std::pair` into the map.



**Parameters**

|                 |                                                                                   |
|-----------------|-----------------------------------------------------------------------------------|
| <code>_↔</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
| <code>_X</code> |                                                                                   |

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

**insert\_or\_assign() [1/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Obj>
pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::insert_or_assign (
    const key_type & __k,
    _Obj && __obj) [inline]
```

Attempts to insert or assign a `std::pair` into the map.

**Parameters**

|                    |                                                                         |
|--------------------|-------------------------------------------------------------------------|
| <code>__k</code>   | Key to use for finding a possibly existing pair in the map.             |
| <code>__obj</code> | Argument used to generate the <code>.second</code> for a pair instance. |

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map. If the pair was already in the map, the `.second` of the pair is assigned from `__obj`.

Insertion requires logarithmic time.

**insert\_or\_assign() [2/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Obj>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert_or_assign (
    const_iterator __hint,
    const key_type & __k,
    _Obj && __obj) [inline]
```

Attempts to insert or assign a `std::pair` into the map.

**Parameters**

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__k</code>    | Key to use for finding a possibly existing pair in the map.                |
| <code>__obj</code>  | Argument used to generate the <code>.second</code> for a pair instance.    |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map. If the pair was already in the map, the `.second` of the pair is assigned from `__obj`.

Insertion requires logarithmic time.

**key\_comp()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
key_compare std::map<_Key, _Tp, _Compare, _Alloc>::key_comp () const [inline]
```

Returns the key comparison object out of which the map was constructed.

Referenced by `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp >>>::at()`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp >>>::insert()`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp >>>::operator[]()`.

**lower\_bound()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
template<typename _Kt>
```

```
auto std::map<_Key, _Tp, _Compare, _Alloc>::lower_bound (
    const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

**lower\_bound()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
template<typename _Kt>
```

```
auto std::map<_Key, _Tp, _Compare, _Alloc>::lower_bound (
    const _Kt & __x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
```

```
[inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::lower_bound (
    const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                                                                                   |                                         |
|-----------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Referenced by [std::map<\\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, \\_Tp > > >::at\(\)](#), [std::map<\\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, \\_Tp > > >::insert\(\)](#), [std::map<\\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, \\_Tp > > >::operator\[\]\(\)](#).

**lower\_bound()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::lower_bound (
    const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                                                                                     |                                         |
|-------------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

**Returns**

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**max\_size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map<_Key, _Tp, _Compare, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the map.

**operator=()** [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map & std::map<_Key, _Tp, _Compare, _Alloc >::operator= (
    const map<_Key, _Tp, _Compare, _Alloc > & ) [default]
```

Map assignment operator.

Whether the allocator is copied depends on the allocator traits.

**operator=()** [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map & std::map<_Key, _Tp, _Compare, _Alloc >::operator= (
    initializer_list<value_type > __l) [inline]
```

Map list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills a map with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map & std::map<_Key, _Tp, _Compare, _Alloc >::operator= (
    map<_Key, _Tp, _Compare, _Alloc > && ) [default]
```

Move assignment operator.

**operator[]()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type & std::map<_Key, _Tp, _Compare, _Alloc >::operator[] (
    const key_type & __k) [inline]
```

Subscript ( [] ) access to map data.

**Parameters**

|    |                                             |
|----|---------------------------------------------|
| ↩  | The key for which data should be retrieved. |
| _k |                                             |

**Returns**

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

**rbegin()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

**rbegin()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

**rend()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

**rend()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

**size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the map.

**swap()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::swap (
    map< _Key, _Tp, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another map.

**Parameters**

|                 |                                                |
|-----------------|------------------------------------------------|
| <code>_↔</code> | A map of the same element and allocator types. |
| <code>_X</code> |                                                |

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

**upper\_bound()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map<_Key, _Tp, _Compare, _Alloc >::upper_bound (
    const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))    [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                         |                                         |
|-------------------------|-----------------------------------------|
| <b><code>__x</code></b> | Key of (key, value) pair to be located. |
|-------------------------|-----------------------------------------|

**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map<_Key, _Tp, _Compare, _Alloc >::upper_bound (
    const _Kt & __x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                         |                                         |
|-------------------------|-----------------------------------------|
| <b><code>__x</code></b> | Key of (key, value) pair to be located. |
|-------------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to first iterator greater than key, or end().

**upper\_bound()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::upper_bound (
    const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                         |                                         |
|-------------------------|-----------------------------------------|
| <b><code>__x</code></b> | Key of (key, value) pair to be located. |
|-------------------------|-----------------------------------------|

**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
    const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <a href="#">_x</a> | Key of (key, value) pair to be located. |
|--------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to first iterator greater than key, or end().

**value\_comp()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::map< _Key, _Tp, _Compare, _Alloc >::value_comp () const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

The documentation for this class was generated from the following file:

- [stl\\_map.h](#)

**5.656 std::mask\_array< \_Tp > Class Template Reference**

```
#include <mask_array.h>
```

**Public Types**

- typedef `_Tp` **value\_type**

**Public Member Functions**

- [mask\\_array](#) (const [mask\\_array](#) &)
- template<class `_Dom`>  
void **operator%=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator%=** (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom`>  
void **operator&=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator&=** (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom`>  
void **operator\*=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator\*=** (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom`>  
void **operator+=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator+=** (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom`>  
void **operator-=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator-=** (const [valarray](#)< `_Tp` > &) const

- template<class \_Dom>  
void **operator**/= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**/= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**<<= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**<<= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**= (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Ex>  
void **operator**= (const \_Expr< \_Ex, \_Tp > &\_\_e) const
- void **operator**= (const \_Tp &) const
- **mask\_array** & **operator**= (const **mask\_array** &)
- void **operator**= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**>>= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**>>= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**^= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**^= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**|= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**|= (const valarray< \_Tp > &) const

## Friends

- class valarray< \_Tp >

### 5.656.1 Detailed Description

**template<class \_Tp>**  
**class std::mask\_array< \_Tp >**

Reference to selected subset of an array.

A mask\_array is a reference to the actual elements of an array specified by a bitmask in the form of an array of bool. The way to get a mask\_array is to call operator[](valarray<bool>) on a valarray. The returned mask\_array then permits carrying operations out on the referenced subset of elements in the original valarray.

For example, if a mask\_array is obtained using the array (false, true, false, true) as an argument, the mask array has two elements referring to array[1] and array[3] in the underlying array.

#### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

### 5.656.2 Member Function Documentation

#### **operator%=( )**

```
template<class _Tp>
void std::mask_array< _Tp >::operator%= (
    const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of v.



**operator&=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator&= (
    const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

**operator\*=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator*= (
    const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

**operator+=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator+= (
    const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

**operator-=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator-= (
    const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

**operator/=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator/= (
    const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

**operator<<=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator<<= (
    const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

**operator>>=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator>>= (
    const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.  
References [mask\\_array\(\)](#).

**operator^=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator^= (
    const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

**operator" |=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator|= (
    const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of v.

The documentation for this class was generated from the following files:

- [valarray](#)
- [mask\\_array.h](#)

## 5.657 `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >` Class Template Reference

```
#include <mask_based_range_hashing.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >`:

**Protected Types**

- typedef `Size_Type` **size\_type**

**Protected Member Functions**

- void **notify\_resized** (`size_type` size)
- `size_type` **range\_hash** (`size_type` hash) const
- void **swap** ([mask\\_based\\_range\\_hashing](#) &other)

**5.657.1 Detailed Description**

```
template<typename Size_Type>
```

```
class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >
```

Range hashing policy.

The documentation for this class was generated from the following file:

- [mask\\_based\\_range\\_hashing.hpp](#)

## 5.658 std::match\_results< \_Bi\_iter, \_Alloc > Class Template Reference

```
#include <regex>
```

Inheritance diagram for std::match\_results< \_Bi\_iter, \_Alloc >:



### Public Types

#### 28.10 Public Types

- typedef `sub_match< _Bi_iter >` `value_type`
- typedef const `value_type` & `const_reference`
- typedef `value_type` & `reference`
- typedef `_Base_type::const_iterator` `const_iterator`
- typedef `const_iterator` `iterator`
- typedef `__iter_traits::difference_type` `difference_type`
- typedef `allocator_traits< _Alloc >::size_type` `size_type`
- typedef `_Alloc` `allocator_type`
- typedef `__iter_traits::value_type` `char_type`
- typedef `std::basic_string< char_type >` `string_type`

### Public Member Functions

- template<typename `_Out_iter`>  
`_Out_iter format` (`_Out_iter __out`, const `match_results< _Bi_iter, _Alloc >::char_type` \*`__fmt_first`, const `match_results< _Bi_iter, _Alloc >::char_type` \*`__fmt_last`, `match_flag_type` `__flags`) const
- bool `ready` () const noexcept

#### 28.10.1 Construction, Copying, and Destruction

- `match_results` ()
- `match_results` (const `_Alloc` &`__a`) noexcept
- `match_results` (const `match_results` &)=default
- `match_results` (`match_results` &&) noexcept=default

- `match_results` & `operator=` (const `match_results` &)=default
- `match_results` & `operator=` (`match_results` &&)=default
- `~match_results` ()=default
- `match_results` (const `match_results` &\_\_m, const `_Alloc` &\_\_a)
- `match_results` (`match_results` &&\_\_m, const `_Alloc` &\_\_a) noexcept(noexcept(`_Base_type`(std::move(\_\_m), ←  
\_\_a)))

### 28.10.2 Size

- `size_type` `size` () const noexcept
- `size_type` `max_size` () const noexcept
- bool `empty` () const noexcept

### 28.10.4 Element Access

- `difference_type` `length` (size\_type \_\_sub=0) const
- `difference_type` `position` (size\_type \_\_sub=0) const
- `string_type` `str` (size\_type \_\_sub=0) const
- `const_reference` `operator[]` (size\_type \_\_sub) const
- `const_reference` `prefix` () const
- `const_reference` `suffix` () const
- `const_iterator` `begin` () const noexcept
- `const_iterator` `cbegin` () const noexcept
- `const_iterator` `end` () const noexcept
- `const_iterator` `cend` () const noexcept

### 28.10.5 Formatting

*These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.*

- template<typename `_Out_iter`>  
    `_Out_iter` `format` (`_Out_iter` \_\_out, const char\_type \* \_\_fmt\_first, const char\_type \* \_\_fmt\_last, `match_flag_type` \_\_flags=`regex_constants::format_default`) const
- template<typename `_Out_iter`, typename `_St`, typename `_Sa`>  
    `_Out_iter` `format` (`_Out_iter` \_\_out, const `basic_string`< char\_type, `_St`, `_Sa` > & \_\_fmt, `match_flag_type` \_\_←  
    flags=`regex_constants::format_default`) const
- template<typename `_St`, typename `_Sa`>  
    `basic_string`< char\_type, `_St`, `_Sa` > `format` (const `basic_string`< char\_type, `_St`, `_Sa` > & \_\_fmt, `match_flag_type` \_\_flags=`regex_constants::format_default`) const
- `string_type` `format` (const char\_type \* \_\_fmt, `match_flag_type` \_\_flags=`regex_constants::format_default`) const

### 28.10.6 Allocator

- `allocator_type` `get_allocator` () const noexcept

### 28.10.7 Swap

- void `swap` (`match_results` &\_\_that) noexcept

### Private Member Functions

- constexpr iterator `begin` () noexcept
- constexpr iterator `end` () noexcept
- constexpr `const_reference` `operator[]` (size\_type \_\_n) const noexcept
- constexpr `reference` `operator[]` (size\_type \_\_n) noexcept
- constexpr void `swap` (`vector` &\_\_x) noexcept

## Friends

- `template<typename, typename, typename>`  
class **regex\_iterator**

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _Bi_iter, typename _Alloc>`  
bool **operator==** (const `match_results< _Bi_iter, _Alloc >` &\_\_m1, const `match_results< _Bi_iter, _Alloc >` &\_\_m2)
- `template<typename _Bi_iter, typename _Alloc>`  
void **swap** (`match_results< _Bi_iter, _Alloc >` &\_\_lhs, `match_results< _Bi_iter, _Alloc >` &\_\_rhs) noexcept

### 5.658.1 Detailed Description

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
class std::match_results< _Bi_iter, _Alloc >
```

The results of a match or search operation.

A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template `match_results`.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The `sub_match` object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the `sub_match::matched` member is always true. The `sub_match` object stored at index `n` denotes what matched the marked sub-expression `n` within the matched expression. If the sub-expression `n` participated in a regular expression match then the `sub_match` member `matched` evaluates to true, and members `first` and `second` denote the range of characters [`first`, `second`) which formed that match. Otherwise `matched` is false, and members `first` and `second` point to the end of the sequence that was searched.

Since

C++11

### 5.658.2 Constructor & Destructor Documentation

**match\_results()** [1/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results () [inline]
```

Constructs a default `match_results` container.

**Postcondition**

`size()` returns 0 and `str()` returns an empty string.

**match\_results()** [2/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
    const _Alloc & __a) [inline], [explicit], [noexcept]
```

Constructs a default `match_results` container.

**Postcondition**

`size()` returns 0 and `str()` returns an empty string.

**match\_results()** [3/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results<_Bi_iter, _Alloc >::match_results (
    const match_results<_Bi_iter, _Alloc > & ) [default]
```

Copy constructs a match\_results.

**match\_results()** [4/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results<_Bi_iter, _Alloc >::match_results (
    match_results<_Bi_iter, _Alloc > && ) [default], [noexcept]
```

Move constructs a match\_results.

**~match\_results()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results<_Bi_iter, _Alloc >::~~match_results () [default]
```

Destroys a match\_results object.

**match\_results()** [5/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results<_Bi_iter, _Alloc >::match_results (
    const match_results<_Bi_iter, _Alloc > & __m,
    const _Alloc & __a) [inline]
```

Constructs a default match\_results container.

**Postcondition**

size() returns 0 and str() returns an empty string.

**match\_results()** [6/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results<_Bi_iter, _Alloc >::match_results (
    match_results<_Bi_iter, _Alloc > && __m,
    const _Alloc & __a) [inline], [noexcept]
```

Constructs a default match\_results container.

**Postcondition**

size() returns 0 and str() returns an empty string.

**5.658.3 Member Function Documentation****begin()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results<_Bi_iter, _Alloc >::begin () const [inline], [noexcept]
```

Gets an iterator to the start of the sub\_match collection.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::operator[]` and `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::operator[]`

**cbegin()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results<_Bi_iter, _Alloc >::cbegin () const [inline], [noexcept]
```

Gets an iterator to the start of the sub\_match collection.

**cend()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results<_Bi_iter, _Alloc >::cend () const [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.

**empty()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results<_Bi_iter, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Indicates if the match\_results contains no results.

**Return values**

|              |                                        |
|--------------|----------------------------------------|
| <i>true</i>  | The match_results object is empty.     |
| <i>false</i> | The match_results object is not empty. |

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator> > >< const char * >::prefix()`,  
and `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator> > >< const char * >::suffix()`.

**end()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results<_Bi_iter, _Alloc >::end () const [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator> > >< const char * >::prefix()`,  
and `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator> > >< const char * >::suffix()`.

**format() [1/4]**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter, typename _St, typename _Sa>
_Out_iter std::match_results<_Bi_iter, _Alloc >::format (
    _Out_iter __out,
    const basic_string< char_type, _St, _Sa > & __fmt,
    match_flag_type __flags = regex_constants::format_default) const [inline]
```

**Precondition**

`ready() == true`

**format() [2/4]**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter>
_Out_iter std::match_results<_Bi_iter, _Alloc >::format (
    _Out_iter __out,
    const char_type * __fmt_first,
    const char_type * __fmt_last,
    match_flag_type __flags = regex_constants::format_default) const
```

**Precondition**

`ready() == true`

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, and `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`.

**format() [3/4]**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>>
template<typename _St, typename _Sa>
basic_string<char_type, _St, _Sa> std::match_results<_Bi_iter, _Alloc>::format (
    const basic_string<char_type, _St, _Sa> & __fmt,
    match_flag_type __flags = regex_constants::format_default) const [inline]
```

**Precondition**

`ready() == true`

**format() [4/4]**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>>
string_type std::match_results<_Bi_iter, _Alloc>::format (
    const char_type * __fmt,
    match_flag_type __flags = regex_constants::format_default) const [inline]
```

**Precondition**

`ready() == true`

**get\_allocator()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>>
allocator_type std::match_results<_Bi_iter, _Alloc>::get_allocator () const [inline], [noexcept]
```

Gets a copy of the allocator.

**length()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>>
difference_type std::match_results<_Bi_iter, _Alloc>::length (
    size_type __sub = 0) const [inline]
```

Gets the length of the indicated submatch.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

**Precondition**

`ready() == true`

This function returns the length of the indicated submatch, or the length of the entire match if `__sub` is zero (the default).



**max\_size()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
size_type std::match_results< _Bi_iter, _Alloc >::max_size () const [inline], [noexcept]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or mark\_count() + 1 if a match was successful. Some matches may be empty.

**Returns**

the number of matches found.

**operator=()** [1/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results & std::match_results< _Bi_iter, _Alloc >::operator= (
    const match_results< _Bi_iter, _Alloc > & ) [default]
```

Assigns rhs to \*this.

**operator=()** [2/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results & std::match_results< _Bi_iter, _Alloc >::operator= (
    match_results< _Bi_iter, _Alloc > && ) [default]
```

Move-assigns rhs to \*this.

**operator[]()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::operator[] (
    size_type __sub) const [inline]
```

Gets a sub\_match reference for the match or submatch.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

**Precondition**

ready() == true

This function gets a reference to the indicated submatch, or the entire match if \_\_sub is zero.

If \_\_sub >= size() then this function returns a sub\_match with a special value indicating no submatch.

**position()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results< _Bi_iter, _Alloc >::position (
    size_type __sub = 0) const [inline]
```

Gets the offset of the beginning of the indicated submatch.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

**Precondition**

`ready() == true`

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of `__sub` is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

**prefix()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
const_reference std::match_results<_Bi_iter, _Alloc>::prefix () const [inline]
```

Gets a `sub_match` representing the match prefix.

**Precondition**

`ready() == true`

This function gets a reference to a `sub_match` object representing the part of the target range between the start of the target range and the start of the match.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*`

**ready()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
bool std::match_results<_Bi_iter, _Alloc>::ready () const [inline], [noexcept]
```

Indicates if the `match_results` is ready.

**Return values**

|              |                                                  |
|--------------|--------------------------------------------------|
| <i>true</i>  | The object has a fully-established result state. |
| <i>false</i> | The object is not ready.                         |

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*`  
`std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::operator`  
`std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::prefix()`,  
and `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::suffix`

**size()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
size_type std::match_results<_Bi_iter, _Alloc>::size () const [inline], [noexcept]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

**Returns**

the number of matches found.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*`  
and `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::operator`

**str()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
string_type std::match_results<_Bi_iter, _Alloc>::str (
    size_type __sub = 0) const [inline]
```

Gets the match or submatch converted to a string type.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

**Precondition**

`ready() == true`

This function gets the submatch (or match, if `__sub` is zero) extracted from the target range and converted to the associated string type.

**suffix()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results<_Bi_iter, _Alloc >::suffix () const [inline]
```

Gets a `sub_match` representing the match suffix.

**Precondition**

`ready() == true`

This function gets a reference to a `sub_match` object representing the part of the target range between the end of the match and the end of the target range.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator > > >< const cha`

**swap()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
void std::match_results<_Bi_iter, _Alloc >::swap (
    match_results<_Bi_iter, _Alloc > & __that) [inline], [noexcept]
```

Swaps the contents of two `match_results`.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator > > >< const cha`

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

## 5.659 `__gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash > Struct` Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >`:



### 5.659.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >
```

Base class for conditionally defining a static data member.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 5.660 `__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >` Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >`:



### Static Public Attributes

- static `null_type` `s_null_type`

#### 5.660.1 Detailed Description

```
template<typename Key, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >
```

Specialization that defines a static data member of type `null_type`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 5.661 `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >` Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

### Public Member Functions

- `mem_fun1_ref_t` (`_Ret` (`_Tp::*``__pf`) (`_Arg`))
- `_Ret operator()` (`_Tp &``r`, `_Arg` `x`) `const`

#### 5.661.1 Detailed Description

template<typename `_Ret`, typename `_Tp`, typename `_Arg`>  
class `std::mem_fun1_ref_t`< `_Ret`, `_Tp`, `_Arg` >

One of the [adaptors for member pointers](#).

#### 5.661.2 Member Typedef Documentation

##### `first_argument_type`

typedef `_Tp` `std::binary_function`< `_Tp`, `_Arg`, `_Ret` >::`first_argument_type` [inherited]  
`first_argument_type` is the type of the first argument

##### `result_type`

typedef `_Ret` `std::binary_function`< `_Tp`, `_Arg`, `_Ret` >::`result_type` [inherited]  
`result_type` is the return type

##### `second_argument_type`

typedef `_Arg` `std::binary_function`< `_Tp`, `_Arg`, `_Ret` >::`second_argument_type` [inherited]  
`second_argument_type` is the type of the second argument

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.662 std::mem\_fun1\_t<\_Ret, \_Tp, \_Arg> Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::mem\_fun1\_t<\_Ret, \_Tp, \_Arg>:



### Public Types

- typedef `_Tp *` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

### Public Member Functions

- `mem_fun1_t` (`_Ret` (`_Tp::*` `__pf`) (`_Arg`))
- `_Ret operator()` (`_Tp *` `__p`, `_Arg` `__x`) `const`

#### 5.662.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_t<_Ret, _Tp, _Arg>
```

One of the [adaptors for member pointers](#).

#### 5.662.2 Member Typedef Documentation

##### first\_argument\_type

```
typedef _Tp * std::binary_function<_Tp *, _Arg, _Ret>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### result\_type

```
typedef _Ret std::binary_function<_Tp *, _Arg, _Ret>::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

```
typedef _Arg std::binary_function< _Tp *, _Arg, _Ret >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**5.663 std::mem\_fun\_ref\_t< \_Ret, \_Tp > Class Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::mem\_fun\_ref\_t< \_Ret, \_Tp >:

**Public Types**

- typedef \_Tp [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

**Public Member Functions**

- `mem_fun_ref_t` (\_Ret(\_Tp::\*\_\_pf)())
- `_Ret operator()` (\_Tp &\_\_r) const

**5.663.1 Detailed Description**

```
template<typename _Ret, typename _Tp>
class std::mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

**5.663.2 Member Typedef Documentation****argument\_type**

```
typedef _Tp std::unary_function< _Tp, _Ret >::argument_type [inherited]
```

argument\_type is the type of the argument



**result\_type**

```
typedef _Ret std::unary_function< _Tp, _Ret >::result_type [inherited]
```

result\_type is the return type

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**5.664 std::mem\_fun\_t< \_Ret, \_Tp > Class Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::mem\_fun\_t< \_Ret, \_Tp >:

**Public Types**

- typedef `_Tp *` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

**Public Member Functions**

- `mem_fun_t` (`_Ret` (`_Tp::*`\_\_pf)())
- `_Ret operator()` (`_Tp *`\_\_p) const

**5.664.1 Detailed Description**

```
template<typename _Ret, typename _Tp>
class std::mem_fun_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

**5.664.2 Member Typedef Documentation****argument\_type**

```
typedef _Tp * std::unary_function< _Tp *, _Ret >::argument_type [inherited]
```

argument\_type is the type of the argument

**result\_type**

```
typedef _Ret std::unary_function< _Tp *, _Ret >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**5.665 std::pmr::memory\_resource Class Reference**

```
#include <memory_resource>
```

Inheritance diagram for std::pmr::memory\_resource:

**Public Member Functions**

- **memory\_resource** (const [memory\\_resource](#) &)=default
- void \* **allocate** (size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- void **deallocate** (void \*\_\_p, size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- bool **is\_equal** (const [memory\\_resource](#) &\_\_other) const noexcept
- [memory\\_resource](#) & **operator=** (const [memory\\_resource](#) &)=default

**5.665.1 Detailed Description**

Class `memory_resource`

Since

C++17

The documentation for this class was generated from the following file:

- [memory\\_resource.h](#)

**5.666 std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f > Class Template Reference**

```
#include <random>
```

**Public Types**

- typedef \_UIntType [result\\_type](#)

## Public Member Functions

- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`  
`mersenne_twister_engine` (`_Sseq &__q`)
- `mersenne_twister_engine` (`result_type __sd`)
- `void discard` (`unsigned long long __z`)
- `result_type operator()` ()
- `template<typename _Sseq>`  
`_If_seed_seq<_Sseq>` `seed` (`_Sseq &__q`)
- `template<typename _Sseq>`  
`auto seed` (`_Sseq &__q`) `-> _If_seed_seq<_Sseq>`
- `void seed` (`result_type __sd=default_seed`)

## Static Public Member Functions

- `static constexpr result_type max` ()
- `static constexpr result_type min` ()

## Static Public Attributes

- `static constexpr result_type default_seed`
- `static constexpr result_type initialization_multiplier`
- `static constexpr size_t mask_bits`
- `static constexpr size_t shift_size`
- `static constexpr size_t state_size`
- `static constexpr result_type tempering_b`
- `static constexpr result_type tempering_c`
- `static constexpr result_type tempering_d`
- `static constexpr size_t tempering_l`
- `static constexpr size_t tempering_s`
- `static constexpr size_t tempering_t`
- `static constexpr size_t tempering_u`
- `static constexpr size_t word_size`
- `static constexpr result_type xor_mask`

## Friends

- `template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits>` `& operator<<` (`std::basic_ostream<_CharT, _Traits> &__os, const std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> &__x`)
- `bool operator==` (`const mersenne_twister_engine &__lhs, const mersenne_twister_engine &__rhs`)
- `template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits>` `& operator>>` (`std::basic_istream<_CharT, _Traits> &__is, std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> &__x`)

### 5.666.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f
>
```

A generalized feedback shift register discrete random number generator.

This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.

The best way to use this generator is with the predefined mt19937 class.

This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

#### Template Parameters

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <code>__w</code> | Word size, the number of bits in each element of the state vector. |
| <code>__n</code> | The degree of recursion.                                           |
| <code>__m</code> | The period parameter.                                              |
| <code>__r</code> | The separation point bit index.                                    |
| <code>__a</code> | The last row of the twist matrix.                                  |
| <code>__u</code> | The first right-shift tempering matrix parameter.                  |
| <code>__d</code> | The first right-shift tempering matrix mask.                       |
| <code>__s</code> | The first left-shift tempering matrix parameter.                   |
| <code>__b</code> | The first left-shift tempering matrix mask.                        |
| <code>__t</code> | The second left-shift tempering matrix parameter.                  |
| <code>__c</code> | The second left-shift tempering matrix mask.                       |
| <code>__l</code> | The second right-shift tempering matrix parameter.                 |
| <code>__f</code> | Initialization multiplier.                                         |

Since

C++11

### 5.666.2 Member Typedef Documentation

#### result\_type

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
```

```
typedef _UIntType std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __↵
s, __b, __t, __c, __l, __f >::result_type
The type of the generated random value.
```

### 5.666.3 Constructor & Destructor Documentation

#### mersenne\_twister\_engine()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t↵
__t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, ↵
__l, __f >::mersenne_twister_engine (
    _Sseq & __q) [inline], [explicit]
```

Constructs a mersenne\_twister\_engine random number generator engine seeded from the seed sequence \_\_q.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

### 5.666.4 Member Function Documentation

#### discard()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t↵
__t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
void std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, ↵
__c, __l, __f >::discard (
    unsigned long long __z)
```

Discard a sequence of random numbers.

#### max()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t↵
__t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, ↵
__u, __d, __s, __b, __t, __c, __l, __f >::max () [inline], [static], [constexpr]
```

Gets the largest possible value in the output range.

#### min()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t↵
__t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, ↵
__u, __d, __s, __b, __t, __c, __l, __f >::min () [inline], [static], [constexpr]
```

Gets the smallest possible value in the output range.

## 5.666.5 Friends And Related Symbol Documentation

## operator&lt;&lt;

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> & __x) [friend]
```

Inserts the current state of a % mersenne\_twister\_engine random number generator engine \_\_x into the output stream \_\_os.

## Parameters

|      |                                                             |
|------|-------------------------------------------------------------|
| __os | An output stream.                                           |
| __x  | A % mersenne_twister_engine random number generator engine. |

## Returns

The output stream with the state of \_\_x inserted or in an error state.

## operator==

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
bool operator== (
    const mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> & __lhs,
    const mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> & __rhs) [friend]
```

Compares two % mersenne\_twister\_engine random number generator objects of the same type for equality.

## Parameters

|       |                                                                   |
|-------|-------------------------------------------------------------------|
| __lhs | A % mersenne_twister_engine random number generator object.       |
| __rhs | Another % mersenne_twister_engine random number generator object. |

## Returns

true if the infinite sequences of generated values would be equal, false otherwise.

## operator&gt;&gt;

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
```

```
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __↵
__a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1,
size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1,
__s1, __b1, __t1, __c1, __l1, __f1 > & __x) [friend]
```

Extracts the current state of a % mersenne\_twister\_engine random number generator engine \_\_x from the input stream \_\_is.

#### Parameters

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>__is</code> | An input stream.                                            |
| <code>__x</code>  | A % mersenne_twister_engine random number generator engine. |

#### Returns

The input stream with the state of \_\_x extracted or in an error state.

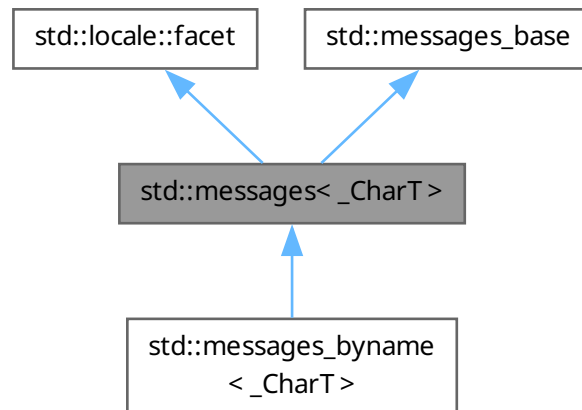
The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 5.667 std::messages<\_CharT> Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::messages<\_CharT>:



#### Public Types

- typedef int **catalog**

- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

### Public Member Functions

- `messages` (`__c_locale __cloc`, `const char *__s`, `size_t __refs=0`)
- `messages` (`size_t __refs=0`)
- void `close` (`catalog __c`) `const`
- `string_type get` (`catalog __c`, `int __set`, `int __msgid`, `const string_type &__s`) `const`
- `catalog open` (`const basic_string<char> &`, `const locale &`, `const char *`) `const`
- `catalog open` (`const basic_string<char> &__s`, `const locale &__loc`) `const`

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual `~messages` ()
- `string_type _M_convert_from_char` (`char *`) `const`
- `char * _M_convert_to_char` (`const string_type &__msg`) `const`
- virtual void `do_close` (`catalog`) `const`
- void `do_close` (`catalog`) `const`
- void `do_close` (`catalog`) `const`
- `string do_get` (`catalog`, `int`, `int`, `const string &`) `const`
- virtual `string_type do_get` (`catalog`, `int`, `int`, `const string_type &__default`) `const`
- `wstring do_get` (`catalog`, `int`, `int`, `const wstring &`) `const`
- virtual `catalog do_open` (`const basic_string<char> &`, `const locale &`) `const`
- `messages<char>::catalog do_open` (`const basic_string<char> &`, `const locale &`) `const`
- `messages<wchar_t>::catalog do_open` (`const basic_string<char> &`, `const locale &`) `const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale &__cloc`) `throw ()`
- static void `_S_create_c_locale` (`__c_locale &__cloc`, `const char *__s`, `__c_locale __old=0`)
- static void `_S_destroy_c_locale` (`__c_locale &__cloc`)
- static `__c_locale _S_get_c_locale` ()
- static `const char * _S_get_c_name` () `throw ()`
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale __cloc`, `const char *__s`)

### Protected Attributes

- `__c_locale _M_c_locale_messages`
- `const char * _M_name_messages`

#### 5.667.1 Detailed Description

`template<typename _CharT>`  
`class std::messages<_CharT>`

Primary class template `messages`.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.



This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar\_t instantiations.

The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

### 5.667.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT>
typedef _CharT std::messages< _CharT >::char_type
Public typedefs.
```

#### string\_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::messages< _CharT >::string_type
Public typedefs.
```

### 5.667.3 Constructor & Destructor Documentation

#### messages() [1/2]

```
template<typename _CharT>
std::messages< _CharT >::messages (
    size_t __refs = 0) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

#### messages() [2/2]

```
template<typename _CharT>
std::messages< _CharT >::messages (
    __c_locale __cloc,
    const char * __s,
    size_t __refs = 0) [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

##### Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__cloc</code> | The C locale.                       |
| <code>__s</code>    | The name of a locale.               |
| <code>__refs</code> | RefCount to pass to the base class. |

References [std::locale::facet::facet\(\)](#).

**~messages()**

```
template<typename _CharT>
std::messages< _CharT >::~~messages () [protected], [virtual]
Destructor.
```

**5.667.4 Member Function Documentation****do\_get()**

```
string std::messages< char >::do_get (
    catalog ,
    int ,
    int ,
    const string & ) const [protected]
```

Specializations for required instantiations.

**5.667.5 Member Data Documentation****id**

```
template<typename _CharT>
locale::id std::messages< _CharT >::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

**5.668 std::messages\_base Struct Reference**

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::messages\_base:



## Public Types

- typedef int **catalog**

### 5.668.1 Detailed Description

Messages facet base class providing catalog typedef.

The documentation for this struct was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

### 5.669 std::messages\_byname<\_CharT> Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::messages\_byname<\_CharT>:



## Public Types

- typedef int **catalog**
- typedef \_CharT **char\_type**
- typedef [basic\\_string](#)<\_CharT> **string\_type**

## Public Member Functions

- **messages\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **messages\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- [string\\_type](#) **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const [string\\_type](#) &\_\_s) const
- catalog **open** (const [basic\\_string](#)<char> &, const [locale](#) &, const char \*) const
- catalog **open** (const [basic\\_string](#)<char> &\_\_s, const [locale](#) &\_\_loc) const

## Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- [string\\_type \\_M\\_convert\\_from\\_char](#) (char \*) const
- char \* [\\_M\\_convert\\_to\\_char](#) (const [string\\_type](#) &\_\_msg) const
- virtual void [do\\_close](#) (catalog) const
- void [do\\_close](#) (catalog) const
- void [do\\_close](#) (catalog) const
- [string do\\_get](#) (catalog, int, int, const [string](#) &) const
- virtual [string\\_type do\\_get](#) (catalog, int, int, const [string\\_type](#) &\_\_dfault) const
- [wstring do\\_get](#) (catalog, int, int, const [wstring](#) &) const
- virtual catalog [do\\_open](#) (const [basic\\_string](#)< char > &, const [locale](#) &) const
- [messages](#)< char >::catalog [do\\_open](#) (const [basic\\_string](#)< char > &, const [locale](#) &) const
- [messages](#)< wchar\_t >::catalog [do\\_open](#) (const [basic\\_string](#)< char > &, const [locale](#) &) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_messages](#)
- const char \* [\\_M\\_name\\_messages](#)

#### 5.669.1 Detailed Description

```
template<typename _CharT>
class std::messages_byname<_CharT>
```

class messages\_byname [22.2.7.2].

#### 5.669.2 Member Function Documentation

##### do\_get()

```
string std::messages< char >::do_get (
    catalog ,
    int ,
    int ,
    const string & ) const [protected], [inherited]
```

Specializations for required instantiations.

#### 5.669.3 Member Data Documentation

##### id

```
template<typename _CharT>
locale::id std::messages<_CharT>::id [static], [inherited]
```

Numpunct facet id.

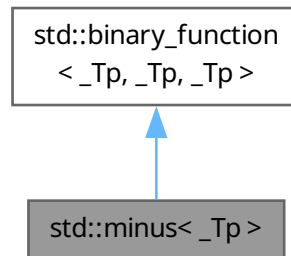
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 5.670 std::minus<\_Tp> Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::minus<\_Tp>:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

#### 5.670.1 Detailed Description

```
template<typename _Tp>
struct std::minus<_Tp>
```

One of the [math functors](#).

#### 5.670.2 Member Typedef Documentation

##### first\_argument\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### result\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type [inherited]
result_type is the return type
```

##### second\_argument\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.671 `std::minus< void >` Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::minus< void >`:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` [is\\_transparent](#)
- typedef void [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename `_Tp`, typename `_Up`>  
constexpr auto **operator()** (`_Tp` &&`__t`, `_Up` &&`__u`) const noexcept(noexcept([std::forward](#)< `_Tp` >(`__t`) - [std::forward](#)< `_Up` >(`__u`))) -> decltype([std::forward](#)< `_Tp` >(`__t`) - [std::forward](#)< `_Up` >(`__u`))
- constexpr void **operator()** (const void &`__x`, const void &`__y`) const

#### 5.671.1 Detailed Description

One of the [math functors](#).

#### 5.671.2 Member Typedef Documentation

##### `first_argument_type`

typedef void [std::binary\\_function](#)< void, void, void >::`first_argument_type`  
`first_argument_type` is the type of the first argument

##### `result_type`

typedef void [std::binary\\_function](#)< void, void, void >::`result_type`  
`result_type` is the return type

##### `second_argument_type`

typedef void [std::binary\\_function](#)< void, void, void >::`second_argument_type`  
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.672 `__gnu_pbds::detail::mod_based_range_hashing< Size_Type >` Class Template Reference

```
#include <mod_based_range_hashing.hpp>
```

### Protected Types

- typedef `Size_Type` `size_type`

### Protected Member Functions

- void `notify_resized` (`size_type` s)
- `size_type` `range_hash` (`size_type` s) const
- void `swap` (`mod_based_range_hashing` &other)

#### 5.672.1 Detailed Description

```
template<typename Size_Type>
```

```
class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >
```

Mod based range hashing.

The documentation for this class was generated from the following file:

- [mod\\_based\\_range\\_hashing.hpp](#)

## 5.673 `std::modulus< _Tp >` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::modulus< _Tp >`:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `_Tp` `result_type`
- typedef `_Tp` `second_argument_type`

**Public Member Functions**

- constexpr **operator**() (const \_Tp &\_\_x, const \_Tp &\_\_y) const

**5.673.1 Detailed Description**

```
template<typename _Tp>
struct std::modulus<_Tp>
```

One of the [math functors](#).

**5.673.2 Member Typedef Documentation****first\_argument\_type**

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

**result\_type**

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

**5.674 std::modulus< void > Struct Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::modulus< void >:

**Public Types**

- typedef void [first\\_argument\\_type](#)
- typedef \_\_is\_transparent [is\\_transparent](#)



- typedef void [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t) %  
[std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t) % [std::forward](#)< \_Up >(\_\_u))
- constexpr void **operator()** (const void &\_\_x, const void &\_\_y) const

### 5.674.1 Detailed Description

One of the [math functors](#).

### 5.674.2 Member Typedef Documentation

#### **first\_argument\_type**

typedef void [std::binary\\_function](#)< void, void, void >::first\_argument\_type  
first\_argument\_type is the type of the first argument

#### **result\_type**

typedef void [std::binary\\_function](#)< void, void, void >::result\_type  
result\_type is the return type

#### **second\_argument\_type**

typedef void [std::binary\\_function](#)< void, void, void >::second\_argument\_type  
second\_argument\_type is the type of the second argument  
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.675 std::money\_base Class Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::money\_base:



### Public Types

- enum { **\_S\_minus** , **\_S\_zero** , **\_S\_end** }
- enum **part** {  
  **none** , **space** , **symbol** , **sign** ,  
  **value** }

### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

### Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**

#### 5.675.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

See also

money\_punct::pos\_format() and money\_punct::neg\_format() for details of how these fields are interpreted.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.676 std::money\_get<\_CharT, \_InIter > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::money\_get<\_CharT, \_InIter >:



### Public Types

- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`
- typedef `basic_string<_CharT >` `string_type`

### Public Member Functions

- `money_get` (`size_t` \_\_refs=0)
- template<bool \_Intl>  
`_GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11` `_M_extract` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `string` &\_\_units) const
- `iter_type get` (`iter_type` \_\_s, `iter_type` \_\_end, bool \_\_intl, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, long double &\_\_units) const
- `iter_type get` (`iter_type` \_\_s, `iter_type` \_\_end, bool \_\_intl, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `string_type` &\_\_digits) const

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual `~money_get` ()
- template<bool \_Intl>  
`iter_type` `_M_extract` (`iter_type` \_\_s, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `string` &\_\_digits) const
- virtual `iter_type do_get` (`iter_type` \_\_s, `iter_type` \_\_end, bool \_\_intl, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, long double &\_\_units) const
- virtual `iter_type do_get` (`iter_type` \_\_s, `iter_type` \_\_end, bool \_\_intl, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `string_type` &\_\_digits) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 5.676.1 Detailed Description

template<typename \_CharT, typename \_InIter>  
class std::money\_get< \_CharT, \_InIter >

Primary class template money\_get.

This facet encapsulates the code to parse and return a monetary amount from a string.

The money\_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money\_get facet.

#### 5.676.2 Member Typedef Documentation

##### char\_type

```
template<typename _CharT, typename _InIter>
typedef _CharT std::money_get< _CharT, _InIter >::char_type
```

Public typedefs.

##### iter\_type

```
template<typename _CharT, typename _InIter>
typedef _InIter std::money_get< _CharT, _InIter >::iter_type
```

Public typedefs.

##### string\_type

```
template<typename _CharT, typename _InIter>
typedef basic_string<_CharT> std::money_get< _CharT, _InIter >::string_type
```

Public typedefs.

#### 5.676.3 Constructor & Destructor Documentation

##### money\_get()

```
template<typename _CharT, typename _InIter>
std::money_get< _CharT, _InIter >::money_get (
    size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

**~money\_get()**

```
template<typename _CharT, typename _InIter>
virtual std::money_get< _CharT, _InIter >::~~money_get () [inline], [protected], [virtual]
Destructor.
```

**5.676.4 Member Function Documentation****do\_get()** [1/2]

```
template<typename _CharT, typename _InIter>
_InIter std::money_get< _CharT, _InIter >::do_get (
    iter_type __s,
    iter_type __end,
    bool __intl,
    ios_base & __io,
    ios_base::iostate & __err,
    long double & __units) const [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for details.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::c\\_str\(\)](#).

Referenced by [get\(\)](#), and [get\(\)](#).

**do\_get()** [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::money_get< _CharT, _InIter >::do_get (
    iter_type __s,
    iter_type __end,
    bool __intl,
    ios_base & __io,
    ios_base::iostate & __err,
    string_type & __digits) const [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for details.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::widen\(\)](#).

**get()** [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::money_get< _CharT, _InIter >::get (
    iter_type __s,
    iter_type __end,
    bool __intl,
    ios_base & __io,
```

```
ios_base::iostate & __err,
long double & __units) const [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `units` as an integral value `moneypunct::frac_digits() * the actual amount`. For example, the string \$10.01 in a US locale would store 1001 in `units`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`. `units` is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

#### Parameters

|                      |                                                                             |
|----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>     | Start of characters to parse.                                               |
| <code>__end</code>   | End of characters to parse.                                                 |
| <code>__intl</code>  | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>    | Source of facets and io state.                                              |
| <code>__err</code>   | Error field to set if parsing fails.                                        |
| <code>__units</code> | Place to store result of parsing.                                           |

#### Returns

Iterator referencing first character beyond valid money amount.

References `do_get()`.

#### get() [2/2]

```
template<typename _CharT, typename _InIter>
iter_type std::money_get< _CharT, _InIter >::get (
    iter_type __s,
    iter_type __end,
    bool __intl,
    ios_base & __io,
    ios_base::iostate & __err,
    string_type & __digits) const [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `digits`. For example, the string \$10.01 in a US locale would store 1001 in `digits`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of `do_get()`.

#### Parameters

|                       |                                                                             |
|-----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>      | Start of characters to parse.                                               |
| <code>__end</code>    | End of characters to parse.                                                 |
| <code>__intl</code>   | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>     | Source of facets and io state.                                              |
| <code>__err</code>    | Error field to set if parsing fails.                                        |
| <code>__digits</code> | Place to store result of parsing.                                           |

## Returns

Iterator referencing first character beyond valid money amount.

References [do\\_get\(\)](#).

## 5.676.5 Member Data Documentation

### id

```
template<typename _CharT, typename _InIter>
locale::id std::money_get< _CharT, _InIter >::id [static]
Numpunct facet id.
```

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.677 std::money\_put< \_CharT, \_OutIter > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for `std::money_put< _CharT, _OutIter >`:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

## Public Member Functions

- [money\\_put](#) (`size_t __refs=0`)
- `template<bool __Intl>`  
`_OutIter M_insert (iter\_type __s, ios\_base &__io, char\_type __fill, const string\_type &__digits) const`
- `iter\_type put (iter\_type __s, bool __intl, ios\_base &__io, char\_type __fill, const string\_type &__digits) const`
- `iter\_type put (iter\_type __s, bool __intl, ios\_base &__io, char\_type __fill, long double __units) const`

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~money\\_put](#) ()
- template<bool \_\_Intl>  
[iter\\_type](#) [\\_M\\_insert](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const [string\\_type](#) &\_\_digits) const
- virtual [iter\\_type](#) [do\\_put](#) ([iter\\_type](#) \_\_s, bool \_\_intl, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const [string\\_type](#) &\_\_digits) const
- virtual [iter\\_type](#) [do\\_put](#) ([iter\\_type](#) \_\_s, bool \_\_intl, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, long double \_\_units) const

## Static Protected Member Functions

- static [\\_\\_c\\_locale](#) [\\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc, const char \*\_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc)
- static [\\_\\_c\\_locale](#) [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale](#) [\\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) \_\_cloc, const char \*\_\_s)

### 5.677.1 Detailed Description

template<typename \_CharT, typename \_OutIter>

class std::money\_put<\_CharT, \_OutIter>

Primary class template money\_put.

This facet encapsulates the code to format and output a monetary amount.

The money\_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money\_put facet.

### 5.677.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::money_put<_CharT, _OutIter>::char_type
```

Public typedefs.

#### iter\_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::money_put<_CharT, _OutIter>::iter_type
```

Public typedefs.

#### string\_type

```
template<typename _CharT, typename _OutIter>
typedef basic_string<_CharT> std::money_put<_CharT, _OutIter>::string_type
```

Public typedefs.



### 5.677.3 Constructor & Destructor Documentation

#### money\_put()

```
template<typename _CharT, typename _OutIter>
std::money_put< _CharT, _OutIter >::money_put (
    size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

#### ~money\_put()

```
template<typename _CharT, typename _OutIter>
virtual std::money_put< _CharT, _OutIter >::~money_put () [inline], [protected], [virtual]
Destructor.
```

### 5.677.4 Member Function Documentation

#### do\_put() [1/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::money_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    bool __intl,
    ios_base & __io,
    char_type __fill,
    const string_type & __digits) const [protected], [virtual]
```

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the string `1001` in a US locale would write `$10.01` to `__s`.

This function is a hook for derived classes to change the value returned.

#### See also

`put()`.

#### Parameters

|                       |                                                                            |
|-----------------------|----------------------------------------------------------------------------|
| <code>__s</code>      | The stream to write to.                                                    |
| <code>__intl</code>   | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> . |
| <code>__io</code>     | Source of facets and io state.                                             |
| <code>__fill</code>   | <code>char_type</code> to use for padding.                                 |
| <code>__digits</code> | Place to store result of parsing.                                          |

#### Returns

Iterator after writing.

**do\_put()** [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::money_put<_CharT, _OutIter >::do_put (
    iter_type __s,
    bool __intl,
    ios_base & __io,
    char_type __fill,
    long double __units) const [protected], [virtual]
```

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to \_\_s. For example, the value 1001 in a US locale would write \$10.01 to \_\_s.

This function is a hook for derived classes to change the value returned.

See also

put().

**Parameters**

|                      |                                                  |
|----------------------|--------------------------------------------------|
| <code>__s</code>     | The stream to write to.                          |
| <code>__intl</code>  | Parameter to use_facet<moneypunct<CharT,intl> >. |
| <code>__io</code>    | Source of facets and io state.                   |
| <code>__fill</code>  | char_type to use for padding.                    |
| <code>__units</code> | Place to store result of parsing.                |

**Returns**

Iterator after writing.

References [std::ios\\_base::getloc\(\)](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base<\\_CharT >::widen\(\)](#).

Referenced by [put\(\)](#), and [put\(\)](#).

**put()** [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::money_put<_CharT, _OutIter >::put (
    iter_type __s,
    bool __intl,
    ios_base & __io,
    char_type __fill,
    const string_type & __digits) const [inline]
```

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to \_\_s. For example, the string 1001 in a US locale would write \$10.01 to \_\_s.

This function works by returning the result of do\_put().

**Parameters**

|                       |                                                  |
|-----------------------|--------------------------------------------------|
| <code>__s</code>      | The stream to write to.                          |
| <code>__intl</code>   | Parameter to use_facet<moneypunct<CharT,intl> >. |
| <code>__io</code>     | Source of facets and io state.                   |
| <code>__fill</code>   | char_type to use for padding.                    |
| <code>__digits</code> | Place to store result of parsing.                |

**Returns**

Iterator after writing.

References [do\\_put\(\)](#).

**put()** [2/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::money_put< _CharT, _OutIter >::put (
    iter_type __s,
    bool __intl,
    ios_base & __io,
    char_type __fill,
    long double __units) const [inline]
```

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the value 1001 in a US locale would write \$10.01 to `__s`. This function works by returning the result of `do_put()`.

**Parameters**

|                      |                                                                             |
|----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>     | The stream to write to.                                                     |
| <code>__intl</code>  | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>    | Source of facets and io state.                                              |
| <code>__fill</code>  | <code>char_type</code> to use for padding.                                  |
| <code>__units</code> | Place to store result of parsing.                                           |

**Returns**

Iterator after writing.

References [do\\_put\(\)](#).

**5.677.5 Member Data Documentation****id**

```
template<typename _CharT, typename _OutIter>
locale::id std::money_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

**5.678 std::moneypunct< \_CharT, \_Intl > Class Template Reference**

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::moneypunct< \_CharT, \_Intl >:



### Public Types

- enum { **\_S\_minus** , **\_S\_zero** , **\_S\_end** }
- typedef \_\_moneypunct\_cache< \_CharT, \_Intl > \_\_cache\_type
- enum part {  
    **none** , **space** , **symbol** , **sign** ,  
    **value** }
- typedef \_CharT char\_type
- typedef basic\_string< \_CharT > string\_type

### Public Member Functions

- `moneypunct` (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
- `moneypunct` (\_\_cache\_type \*\_\_cache, size\_t \_\_refs=0)
- `moneypunct` (size\_t \_\_refs=0)
- `string_type curr_symbol` () const
- `char_type decimal_point` () const
- `int frac_digits` () const
- `string grouping` () const
- `string_type negative_sign` () const
- `string_type positive_sign` () const
- `char_type thousands_sep` () const
- pattern `pos_format` () const
- pattern `neg_format` () const

### Static Public Member Functions

- static pattern `_S_construct_pattern` (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

### Static Public Attributes

- static const char \* `_S_atoms`
- static const pattern `_S_default_pattern`
- static `locale::id` `id`
- static const bool `intl`

### Protected Member Functions

- virtual `~moneypunct` ()
- void `_M_initialize_moneypunct` (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- void `_M_initialize_moneypunct` (\_\_c\_locale, const char \*)
- void `_M_initialize_moneypunct` (\_\_c\_locale, const char \*)
- void `_M_initialize_moneypunct` (\_\_c\_locale, const char \*)
- void `_M_initialize_moneypunct` (\_\_c\_locale, const char \*)
- virtual `string_type do_curr_symbol` () const
- virtual `char_type do_decimal_point` () const
- virtual int `do_frac_digits` () const
- virtual `string do_grouping` () const
- virtual pattern `do_neg_format` () const
- virtual `string_type do_negative_sign` () const
- virtual pattern `do_pos_format` () const
- virtual `string_type do_positive_sign` () const
- virtual `char_type do_thousands_sep` () const

### Static Protected Member Functions

- static \_\_c\_locale `_S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static \_\_c\_locale `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 5.678.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct< _CharT, _Intl >
```

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

#### 5.678.2 Member Typedef Documentation

##### `char_type`

```
template<typename _CharT, bool _Intl>
typedef _CharT std::moneypunct< _CharT, _Intl >::char_type
```

Public typedefs.

**string\_type**

```
template<typename _CharT, bool _Intl>
typedef basic\_string<_CharT> std::moneypunct<_CharT, _Intl>::string_type
Public typedefs.
```

**5.678.3 Constructor & Destructor Documentation****moneypunct()** [1/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct<_CharT, _Intl>::moneypunct (
    size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

**moneypunct()** [2/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct<_CharT, _Intl>::moneypunct (
    __cache_type * __cache,
    size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is an internal constructor.

**Parameters**

|                      |                                 |
|----------------------|---------------------------------|
| <code>__cache</code> | Cache for optimization.         |
| <code>__refs</code>  | Passed to the base facet class. |

References [std::locale::facet::facet\(\)](#).

**moneypunct()** [3/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct<_CharT, _Intl>::moneypunct (
    __c_locale __cloc,
    const char * __s,
    size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | The C locale.                   |
| <code>__s</code>    | The name of a locale.           |
| <code>__refs</code> | Passed to the base facet class. |

References [std::locale::facet::facet\(\)](#).

**~moneypunct()**

```
template<typename _CharT, bool _Intl>
virtual std::moneypunct< _CharT, _Intl >::~~moneypunct () [protected], [virtual]
Destructor.
```

**5.678.4 Member Function Documentation****curr\_symbol()**

```
template<typename _CharT, bool _Intl>
string\_type std::moneypunct< _CharT, _Intl >::curr_symbol () const [inline]
```

Return currency symbol string.

This function returns a [string\\_type](#) to use as a currency symbol. It does so by returning [moneypunct](#)<[char\\_type](#)>::do\_curr\_symbol().

**Returns**

*string\_type* representing a currency symbol.

References [do\\_curr\\_symbol\(\)](#).

**decimal\_point()**

```
template<typename _CharT, bool _Intl>
char\_type std::moneypunct< _CharT, _Intl >::decimal_point () const [inline]
```

Return decimal point character.

This function returns a [char\\_type](#) to use as a decimal point. It does so by returning [moneypunct](#)<[char\\_type](#)>::do\_decimal\_point().

**Returns**

*char\_type* representing a decimal point.

References [do\\_decimal\\_point\(\)](#).

**do\_curr\_symbol()**

```
template<typename _CharT, bool _Intl>
virtual string\_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const [inline], [protected], [virtual]
```

Return currency symbol string.

This function returns a [string\\_type](#) to use as a currency symbol. This function is a hook for derived classes to change the value returned.

**See also**

[curr\\_symbol\(\)](#) for details.

**Returns**

*string\_type* representing a currency symbol.

Referenced by [curr\\_symbol\(\)](#).

**do\_decimal\_point()**

```
template<typename _CharT, bool _Intl>
virtual char\_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const [inline], [protected], [virtual]
```

Return decimal point character.

Returns a [char\\_type](#) to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Referenced by [decimal\\_point\(\)](#).

**do\_frac\_digits()**

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct<\_CharT, \_Intl>::do\_frac\_digits () const [inline], [protected],
[virtual]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

**See also**

[frac\\_digits\(\)](#) for details.

**Returns**

Number of digits in amount fraction.

Referenced by [frac\\_digits\(\)](#).

**do\_grouping()**

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct<\_CharT, \_Intl>::do\_grouping () const [inline], [protected],
[virtual]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Referenced by [grouping\(\)](#).

**do\_neg\_format()**

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct<\_CharT, \_Intl>::do\_neg\_format () const [inline], [protected],
[virtual]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[neg\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Referenced by [neg\\_format\(\)](#).



**do\_negative\_sign()**

```
template<typename _CharT, bool _Intl>
virtual string\_type std::moneypunct< _CharT, _Intl >::do_negative_sign () const [inline], [protected],
[virtual]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

**See also**

`negative_sign()` for details.

**Returns**

*string\_type* representing a negative sign.

Referenced by [negative\\_sign\(\)](#).

**do\_pos\_format()**

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const [inline], [protected],
[virtual]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

`pos_format()` for details.

**Returns**

Pattern for money values.

Referenced by [pos\\_format\(\)](#).

**do\_positive\_sign()**

```
template<typename _CharT, bool _Intl>
virtual string\_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const [inline], [protected],
[virtual]
```

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

**See also**

`positive_sign()` for details.

**Returns**

*string\_type* representing a positive sign.

Referenced by [positive\\_sign\(\)](#).

**do\_thousands\_sep()**

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct<_CharT, _Intl>::do_thousands_sep () const [inline], [protected], [virtual]
```

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Referenced by [thousands\\_sep\(\)](#).

**frac\_digits()**

```
template<typename _CharT, bool _Intl>
int std::moneypunct<_CharT, _Intl>::frac_digits () const [inline]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `returning moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

**Returns**

Number of digits in amount fraction.

References [do\\_frac\\_digits\(\)](#).

**grouping()**

```
template<typename _CharT, bool _Intl>
string std::moneypunct<_CharT, _Intl>::grouping () const [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was 32, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

References [do\\_grouping\(\)](#).

**neg\_format()**

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct<_CharT, _Intl>::neg_format () const [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `returning moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field. The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

#### Returns

Pattern for money values.

References [do\\_neg\\_format\(\)](#).

### negative\_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::negative_sign () const [inline]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a negative sign.

References [do\\_negative\\_sign\(\)](#).

### pos\_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::pos_format () const [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

#### Returns

Pattern for money values.

References [do\\_pos\\_format\(\)](#).

### positive\_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const [inline]
```

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `money_punct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a positive sign.

References [do\\_positive\\_sign\(\)](#).

#### `thousands_sep()`

```
template<typename _CharT, bool _Intl>
```

```
char_type std::money_punct<_CharT, _Intl>::thousands_sep () const [inline]
```

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `money_punct<char_type>::do_thousands_sep()`.

#### Returns

`char_type` representing a thousands separator.

References [do\\_thousands\\_sep\(\)](#).

### 5.678.5 Member Data Documentation

#### `id`

```
template<typename _CharT, bool _Intl>
```

```
locale::id std::money_punct<_CharT, _Intl>::id [static]
```

Numpunct facet id.

#### `intl`

```
template<typename _CharT, bool _Intl>
```

```
const bool std::money_punct<_CharT, _Intl>::intl [static]
```

This value is provided by the standard, but no reason for its existence.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

### 5.679 `std::moneypunct_byname<_CharT, _Intl>` Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for `std::moneypunct_byname<_CharT, _Intl>`:



### Public Types

- enum { **\_S\_minus** , **\_S\_zero** , **\_S\_end** }
- typedef `__moneypunct_cache<_CharT, _Intl>` **\_\_cache\_type**
- typedef `_CharT` **char\_type**
- enum **part** {  
    **none** , **space** , **symbol** , **sign** ,  
    **value** }
- typedef `basic_string<_CharT>` **string\_type**

### Public Member Functions

- **moneypunct\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **moneypunct\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- [string\\_type](#) **curr\_symbol** () const
- char\_type **decimal\_point** () const
- int **frac\_digits** () const
- [string](#) **grouping** () const
- [string\\_type](#) **negative\_sign** () const
- [string\\_type](#) **positive\_sign** () const
- char\_type **thousands\_sep** () const
- pattern [pos\\_format](#) () const
- pattern [neg\\_format](#) () const

### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

### Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**
- static [locale::id](#) **id**
- static const bool **intl**

### Protected Member Functions

- void **\_M\_initialize\_moneypunct** (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- virtual [string\\_type](#) **do\_curr\_symbol** () const
- virtual char\_type **do\_decimal\_point** () const
- virtual int **do\_frac\_digits** () const
- virtual [string](#) **do\_grouping** () const
- virtual pattern **do\_neg\_format** () const
- virtual [string\\_type](#) **do\_negative\_sign** () const
- virtual pattern **do\_pos\_format** () const
- virtual [string\\_type](#) **do\_positive\_sign** () const
- virtual char\_type **do\_thousands\_sep** () const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 5.679.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct_byname< _CharT, _Intl >
```

class moneypunct\_byname [22.2.6.4].

#### 5.679.2 Member Function Documentation

##### curr\_symbol()

```
template<typename _CharT, bool _Intl>
string\_type std::moneypunct< _CharT, _Intl >::curr_symbol () const [inline], [inherited]
```

Return currency symbol string.

This function returns a [string\\_type](#) to use as a currency symbol. It does so by returning returning moneypunct<char↵\_type>::do\_curr\_symbol().

##### Returns

*string\_type* representing a currency symbol.

References [do\\_curr\\_symbol\(\)](#).

**decimal\_point()**

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point () const [inline], [inherited]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `returning moneypunct<char_↵type>::do_decimal_point()`.

**Returns**

*char\_type* representing a decimal point.

References [do\\_decimal\\_point\(\)](#).

**do\_curr\_symbol()**

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const [inline], [protected],
[virtual], [inherited]
```

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

**See also**

`curr_symbol()` for details.

**Returns**

*string\_type* representing a currency symbol.

Referenced by [curr\\_symbol\(\)](#).

**do\_decimal\_point()**

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Referenced by [decimal\\_point\(\)](#).

**do\_frac\_digits()**

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits () const [inline], [protected],
[virtual], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

**See also**

`frac_digits()` for details.

### Returns

Number of digits in amount fraction.

Referenced by [frac\\_digits\(\)](#).

### `do_grouping()`

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct<_CharT, _Intl >::do_grouping () const [inline], [protected],
[virtual], [inherited]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

### See also

[grouping\(\)](#) for details.

### Returns

String representing grouping specification.

Referenced by [grouping\(\)](#).

### `do_neg_format()`

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct<_CharT, _Intl >::do_neg_format () const [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

### See also

[neg\\_format\(\)](#) for details.

### Returns

Pattern for money values.

Referenced by [neg\\_format\(\)](#).

### `do_negative_sign()`

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct<_CharT, _Intl >::do_negative_sign () const [inline], [protected],
[virtual], [inherited]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

### See also

[negative\\_sign\(\)](#) for details.

### Returns

*string\_type* representing a negative sign.

Referenced by [negative\\_sign\(\)](#).



**do\_pos\_format()**

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[pos\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Referenced by [pos\\_format\(\)](#).

**do\_positive\_sign()**

```
template<typename _CharT, bool _Intl>
virtual string\_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const [inline], [protected],
[virtual], [inherited]
```

Return positive sign string.

This function returns a [string\\_type](#) to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

**See also**

[positive\\_sign\(\)](#) for details.

**Returns**

*string\_type* representing a positive sign.

Referenced by [positive\\_sign\(\)](#).

**do\_thousands\_sep()**

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep () const [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.

Returns a [char\\_type](#) to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Referenced by [thousands\\_sep\(\)](#).

**frac\_digits()**

```
template<typename _CharT, bool _Intl>
int std::moneypunct< _CharT, _Intl >::frac_digits () const [inline], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning [std::moneypunct<char\\_type>::do\\_frac\\_digits\(\)](#).

The fractional part of a money amount is optional. But if it is present, there must be [frac\\_digits\(\)](#) digits.

**Returns**

Number of digits in amount fraction.

References [do\\_frac\\_digits\(\)](#).

**grouping()**

```
template<typename _CharT, bool _Intl>
string std::moneypunct<_CharT, _Intl>::grouping () const [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was 32, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

References [do\\_grouping\(\)](#).

**neg\_format()**

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct<_CharT, _Intl>::neg_format () const [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

**Returns**

Pattern for money values.

References [do\\_neg\\_format\(\)](#).

**negative\_sign()**

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct<_CharT, _Intl>::negative_sign () const [inline], [inherited]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a negative sign.

References [do\\_negative\\_sign\(\)](#).

**pos\_format()**

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::pos_format () const [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

**Returns**

Pattern for money values.

References [do\\_pos\\_format\(\)](#).

**positive\_sign()**

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const [inline], [inherited]
```

Return positive sign string.

This function returns a *string\_type* to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a positive sign.

References [do\\_positive\\_sign\(\)](#).

**thousands\_sep()**

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep () const [inline], [inherited]
```

Return thousands separator character.

This function returns a *char\_type* to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

**Returns**

*char\_type* representing a thousands separator.

References [do\\_thousands\\_sep\(\)](#).

### 5.679.3 Member Data Documentation

#### id

```
template<typename _CharT, bool _Intl>
```

```
locale::id std::moneypunct< _CharT, _Intl >::id [static], [inherited]
```

Numpunct facet id.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.680 std::pmr::monotonic\_buffer\_resource Class Reference

```
#include <memory_resource>
```

Inheritance diagram for std::pmr::monotonic\_buffer\_resource:



### Public Member Functions

- **monotonic\_buffer\_resource** (const [monotonic\\_buffer\\_resource](#) &)=delete
- **monotonic\_buffer\_resource** ([memory\\_resource](#) \*\_\_upstream) noexcept
- **monotonic\_buffer\_resource** (size\_t \_\_initial\_size) noexcept
- **monotonic\_buffer\_resource** (size\_t \_\_initial\_size, [memory\\_resource](#) \*\_\_upstream) noexcept
- **monotonic\_buffer\_resource** (void \* \_\_buffer, size\_t \_\_buffer\_size) noexcept
- **monotonic\_buffer\_resource** (void \* \_\_buffer, size\_t \_\_buffer\_size, [memory\\_resource](#) \*\_\_upstream) noexcept
- void \* **allocate** (size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- void **deallocate** (void \* \_\_p, size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- bool **is\_equal** (const [memory\\_resource](#) &\_\_other) const noexcept
- [monotonic\\_buffer\\_resource](#) & **operator=** (const [monotonic\\_buffer\\_resource](#) &)=delete
- void **release** () noexcept
- [memory\\_resource](#) \* **upstream\_resource** () const noexcept

### Protected Member Functions

- void \* **do\_allocate** (size\_t \_\_bytes, size\_t \_\_alignment) override
- void **do\_deallocate** (void \*, size\_t, size\_t) override
- bool **do\_is\_equal** (const [memory\\_resource](#) &\_\_other) const noexcept override

### 5.680.1 Detailed Description

A memory resource that allocates from a fixed-size buffer.

The main feature of a `pmr::monotonic_buffer_resource` is that its `do_deallocate` does nothing. This makes it very fast because there is no need to manage a free list, and every allocation simply returns a new block of memory, rather than searching for a suitably-sized free block. Because deallocating is a no-op, the amount of memory used by the resource only grows until `release()` (or the destructor) is called to return all memory to upstream.

A `monotonic_buffer_resource` can be initialized with a buffer that will be used to satisfy all allocation requests, until the buffer is full. After that a new buffer will be allocated from the upstream resource. By using a stack buffer and `pmr::null_memory_resource()` as the upstream you can get a memory resource that only uses the stack and never dynamically allocates.

Since

C++17

### 5.680.2 Member Function Documentation

#### `do_allocate()`

```
void * std::pmr::monotonic_buffer_resource::do_allocate (
    size_t __bytes,
    size_t __alignment) [inline], [override], [protected], [virtual]
```

Implements [std::pmr::memory\\_resource](#).

#### `do_deallocate()`

```
void std::pmr::monotonic_buffer_resource::do_deallocate (
    void * ,
    size_t ,
    size_t ) [inline], [override], [protected], [virtual]
```

Implements [std::pmr::memory\\_resource](#).

#### `do_is_equal()`

```
bool std::pmr::monotonic_buffer_resource::do_is_equal (
    const memory\_resource & __other) const [inline], [override], [protected], [virtual],
[noexcept]
```

Implements [std::pmr::memory\\_resource](#).

The documentation for this class was generated from the following file:

- [memory\\_resource](#)

## 5.681 `std::move_iterator<_Iterator>` Class Template Reference

```
#include <stl_iterator.h>
```

### Public Types

- using `difference_type`
- using `iterator_concept`
- using `iterator_type`
- using `pointer`
- using `reference`
- using `value_type`

## Public Member Functions

- template<typename \_Iter>  
requires \_\_convertible<\_Iter>  
constexpr **move\_iterator** (const **move\_iterator**<\_Iter> &\_\_i)
- constexpr **move\_iterator** (iterator\_type \_\_i)
- constexpr iterator\_type **base** () &&
- constexpr const iterator\_type & **base** () const &noexcept
- constexpr reference **operator\*** () const
- constexpr **move\_iterator** **operator+** (difference\_type \_\_n) const
- constexpr **move\_iterator** & **operator++** ()
- constexpr **move\_iterator** **operator++** (int)
- constexpr void **operator++** (int)
- constexpr **move\_iterator** & **operator+=** (difference\_type \_\_n)
- constexpr **move\_iterator** **operator-** (difference\_type \_\_n) const
- constexpr **move\_iterator** & **operator--** ()
- constexpr **move\_iterator** **operator--** (int)
- constexpr **move\_iterator** & **operator-=** (difference\_type \_\_n)
- constexpr pointer **operator->** () const
- template<typename \_Iter>  
requires \_\_convertible<\_Iter> && assignable\_from<\_Iterator&, const \_Iter&>  
constexpr **move\_iterator** & **operator=** (const **move\_iterator**<\_Iter> &\_\_i)
- constexpr reference **operator[]** (difference\_type \_\_n) const

## Friends

- constexpr iter\_rvalue\_reference\_t<\_Iterator> **iter\_move** (const **move\_iterator** &\_\_i) noexcept(noexcept(ranges::iter\_move(\_\_i.\_M\_current)))
- template<indirectly\_swappable<\_Iterator> \_Iter2>  
constexpr void **iter\_swap** (const **move\_iterator** &\_\_x, const **move\_iterator**<\_Iter2> &\_\_y) noexcept(noexcept(ranges::iter\_swap(\_\_x.\_M\_current, \_\_y.\_M\_current)))
- template<sized\_sentinel\_for<\_Iterator> \_Sent>  
constexpr iter\_difference\_t<\_Iterator> **operator-** (const **move\_iterator** &\_\_x, const **move\_sentinel**<\_Sent> &\_\_y)
- template<sized\_sentinel\_for<\_Iterator> \_Sent>  
constexpr iter\_difference\_t<\_Iterator> **operator-** (const **move\_sentinel**<\_Sent> &\_\_x, const **move\_iterator** &\_\_y)
- template<sentinel\_for<\_Iterator> \_Sent>  
constexpr bool **operator==** (const **move\_iterator** &\_\_x, const **move\_sentinel**<\_Sent> &\_\_y)

### 5.681.1 Detailed Description

template<typename \_Iterator>  
class std::move\_iterator<\_Iterator>

An iterator adaptor that yields an rvalue reference.

Class template move\_iterator is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 5.682 std::move\_only\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)> Class Template Reference

```
#include <functional>
```

### Public Types

- using **result\_type**

### Public Member Functions

- [move\\_only\\_function](#) () noexcept
- [template<typename \\_Fn, typename \\_Vt = decay\\_t<\\_Fn>>](#)  
requires (is\_same\_v<\_Vt, move\_only\_function>) && (!is\_in\_place\_type\_v<\_Vt>) && \_\_is\_callable\_from<\_Vt>  
[move\\_only\\_function](#) (\_Fn &&\_\_f) noexcept(\_S\_nothrow\_init< \_Vt, \_Fn >())
- [template<typename \\_Tp, typename... \\_Args>](#)  
requires is\_constructible\_v<\_Tp, \_Args...> && \_\_is\_callable\_from<\_Tp>  
[move\\_only\\_function](#) (in\_place\_type\_t< \_Tp >, \_Args &&... \_\_args) noexcept(\_S\_nothrow\_init< \_Tp, \_Args... >())
- [template<typename \\_Tp, typename \\_Up, typename... \\_Args>](#)  
requires is\_constructible\_v<\_Tp, initializer\_list<\_Up>&, \_Args...> && \_\_is\_callable\_from<\_Tp>  
[move\\_only\\_function](#) (in\_place\_type\_t< \_Tp >, initializer\_list< \_Up > \_\_il, \_Args &&... \_\_args) noexcept(\_S\_nothrow\_init< \_Tp, initializer\_list< \_Up > &, \_Args... >())
- [move\\_only\\_function](#) (move\_only\_function &&\_\_x) noexcept
- [move\\_only\\_function](#) (nullptr\_t) noexcept
- [operator bool](#) () const noexcept
- [\\_Res operator\(\)](#) (\_ArgTypes... \_\_args) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)
- [template<typename \\_Fn>](#)  
requires is\_constructible\_v<move\_only\_function, \_Fn>  
[move\\_only\\_function](#) & [operator=](#) (\_Fn &&\_\_f) noexcept(is\_nothrow\_constructible\_v< move\_only\_function, \_Fn >)
- [move\\_only\\_function](#) & [operator=](#) (move\_only\_function &&\_\_x) noexcept
- [move\\_only\\_function](#) & [operator=](#) (nullptr\_t) noexcept
- void [swap](#) (move\_only\_function &\_\_x) noexcept

### Friends

- [template<typename \\_Func>](#)  
auto & [\\_\\_polyfunc::\\_\\_base\\_of](#) (\_Func &) noexcept
- [template<typename \\_Func>](#)  
auto & [\\_\\_polyfunc::\\_\\_invoker\\_of](#) (\_Func &) noexcept
- [template<typename \\_Dst, typename \\_Src>](#)  
constexpr bool [\\_\\_polyfunc::\\_\\_is\\_invoker\\_convertible](#) () noexcept
- bool [operator==](#) (const move\_only\_function &\_\_x, nullptr\_t) noexcept
- void [swap](#) (move\_only\_function &\_\_x, move\_only\_function &\_\_y) noexcept

#### 5.682.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
class std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
```

Polymorphic function wrapper.

Since

C++23

The `std::move_only_function` class template is a call wrapper similar to `std::function`, but does not require the stored target function to be copyable.

It also supports const-qualification, ref-qualification, and no-throw guarantees. The qualifications and exception-specification of the `move_only_function::operator()` member function are respected when invoking the target function.

## 5.682.2 Constructor & Destructor Documentation

### move\_only\_function() [1/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
() [inline], [noexcept]
```

Creates an empty object.

References [move\\_only\\_function\(\)](#).

Referenced by [move\\_only\\_function\(\)](#), [move\\_only\\_function\(\)](#), [move\\_only\\_function\(\)](#), [move\\_only\\_function\(\)](#), [move\\_only\\_function\(\)](#), [move\\_only\\_function\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator==\(\)](#), [swap\(\)](#), and [swap\(\)](#).

### move\_only\_function() [2/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
    nullptr_t ) [inline], [noexcept]
```

Creates an empty object.

References [move\\_only\\_function\(\)](#).

### move\_only\_function() [3/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
    move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Moves the target object, leaving the source empty.

References [move\\_only\\_function\(\)](#).

### move\_only\_function() [4/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = decay_t<_Fn>>
requires (!is_same_v<_Vt, move_only_function>) && (!__is_in_place_type_v<_Vt>) && __is_callable<
_from<_Vt>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
    _Fn && __f) [inline], [noexcept]
```

Stores a target object initialized from the argument.

References [move\\_only\\_function\(\)](#), and [std::forward\(\)](#).

### move\_only\_function() [5/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename... _Args>
```



```
requires is_constructible_v<_Tp, _Args...> && __is_callable_from<_Tp>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
    in_place_type_t< _Tp > ,
    _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.

References [move\\_only\\_function\(\)](#), and [std::forward\(\)](#).

### move\_only\_function() [6/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename _Up, typename... _Args>
requires is_constructible_v<_Tp, initializer_list<_Up>&, _Args...> && __is_callable_from<_Tp>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
    in_place_type_t< _Tp > ,
    initializer_list< _Up > __il,
    _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.

References [move\\_only\\_function\(\)](#), and [std::forward\(\)](#).

## 5.682.3 Member Function Documentation

### operator bool()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator bool ()
const [inline], [explicit], [noexcept]
```

True if a target object is present, false otherwise.

### operator()()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
_Res std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
    _ArgTypes... __args) [inline], [noexcept]
```

Invoke the target object.

The target object will be invoked using the supplied arguments, and as an lvalue or rvalue, and as const or non-const, as dictated by the template arguments of the `move_only_function` specialization.

#### Precondition

Must not be empty.

References [std::forward\(\)](#), and [operator\(\)\(\)](#).

Referenced by [operator\(\)\(\)](#).

### operator=() [1/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn>
requires is_constructible_v<move_only_function, _Fn>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
    _Fn && __f) [inline], [noexcept]
```

Stores a new target object, initialized from the argument.

References [move\\_only\\_function\(\)](#), [std::forward\(\)](#), and [operator=\(\)](#).

### **operator=()** [2/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
    move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
```

[inline], [noexcept]

Stores a new target object, leaving `x` empty.

References [move\\_only\\_function\(\)](#), [std::addressof\(\)](#), and [operator=\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

### **operator=()** [3/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
    nullptr_t ) [inline], [noexcept]
```

Destroys the target object (if any).

References [move\\_only\\_function\(\)](#), and [operator=\(\)](#).

### **swap()**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::swap (
    move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x)
[inline], [noexcept]
```

Exchange the target objects (if any).

References [move\\_only\\_function\(\)](#), and [swap\(\)](#).

Referenced by [swap\(\)](#), and [swap\(\)](#).

## 5.682.4 Friends And Related Symbol Documentation

### **operator==**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
bool operator== (
    const move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
    nullptr_t ) [friend]
```

Check for emptiness by comparing with `nullptr`.

References [move\\_only\\_function\(\)](#), and [operator==](#).

Referenced by [operator==](#).

### **swap**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void swap (
    move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
    move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __y)
[friend]
```

Exchange the target objects (if any).

References [move\\_only\\_function\(\)](#), and [swap\(\)](#).

The documentation for this class was generated from the following file:

- [mofunc\\_impl.h](#)

### 5.683 `std::move_sentinel<_Sent >` Class Template Reference

```
#include <stl_iterator.h>
```

#### Public Member Functions

- constexpr **move\_sentinel** (`_Sent __s`) noexcept(is\_nothrow\_move\_constructible\_v<\_Sent >)
- template<typename `_S2`>  
requires convertible\_to<const `_S2`&, `_Sent`>  
constexpr **move\_sentinel** (const [move\\_sentinel](#)< `_S2` > &`__s`) noexcept(is\_nothrow\_constructible\_v<\_Sent, const `_S2` & >)
- constexpr `_Sent` **base** () const noexcept(is\_nothrow\_copy\_constructible\_v<\_Sent >)
- template<typename `_S2`>  
requires assignable\_from<\_Sent&, const `_S2`&>  
constexpr [move\\_sentinel](#) & **operator=** (const [move\\_sentinel](#)< `_S2` > &`__s`) noexcept(is\_nothrow\_assignable\_v<\_Sent, const `_S2` & >)

#### 5.683.1 Detailed Description

```
template<semiregular _Sent>
class std::move_sentinel<_Sent >
```

A sentinel adaptor for use with `std::move_iterator`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

### 5.684 `std::__debug::multimap<_Key, _Tp, _Compare, _Allocator >` Class Template Reference

```
#include <multimap.h>
```

Inheritance diagram for `std::__debug::multimap<_Key, _Tp, _Compare, _Allocator >`:



#### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< [\\_Base\\_const\\_iterator](#), `multimap` > **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef [std::reverse\\_iterator](#)< [const\\_iterator](#) > **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< [\\_Base\\_iterator](#), `multimap` > **iterator**
- typedef `_Compare` **key\_compare**

- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- using **node\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- **multimap** (`_Base_ref __x`)
- template<typename `_InputIterator`>  
**multimap** (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- template<typename `_InputIterator`>  
**multimap** (`_InputIterator __first`, `_InputIterator __last`, `const allocator_type &__a`)
- **multimap** (`const _Compare &__comp`, `const _Allocator &__a=_Allocator()`)
- **multimap** (`const allocator_type &__a`)
- **multimap** (`const multimap &`)=default
- **multimap** (`const multimap &__m`, `const __type_identity_t< allocator_type > &__a`)
- **multimap** (`initializer_list< value_type > __l`, `const _Compare &__c=_Compare()`, `const allocator_type &__a`, `__a=allocator_type()`)
- **multimap** (`initializer_list< value_type > __l`, `const allocator_type &__a`)
- **multimap** (`multimap &&`)=default
- **multimap** (`multimap &&__m`, `const __type_identity_t< allocator_type > &__a`) noexcept(noexcept(`_Base`(`std::move`(`__m`), `__a`)))
- `__attribute__((abi_tag("cxx11")))` **iterator** `erase(iterator __position)`
- `const _Base & __M_base ()` const noexcept
- `_Base & __M_base ()` noexcept
- `const_iterator begin ()` const noexcept
- **iterator** `begin ()` noexcept
- `const_iterator cbegin ()` const noexcept
- `const_iterator cend ()` const noexcept
- void **clear** () noexcept
- `const_reverse_iterator crbegin ()` const noexcept
- `const_reverse_iterator crend ()` const noexcept
- template<typename... `_Args`>  
**iterator** `emplace (_Args &&... __args)`
- template<typename... `_Args`>  
**iterator** `emplace_hint (const_iterator __pos, _Args &&... __args)`
- `const_iterator end ()` const noexcept
- **iterator** `end ()` noexcept
- template<typename `_Kt`, typename `_Req = typename __has_is_transparent<_Compare, _Kt>::type`>  
`std::pair< iterator, iterator >` **equal\_range** (`const _Kt &__x`)
- template<typename `_Kt`, typename `_Req = typename __has_is_transparent<_Compare, _Kt>::type`>  
`std::pair< const_iterator, const_iterator >` **equal\_range** (`const _Kt &__x`) const
- `std::pair< iterator, iterator >` **equal\_range** (`const key_type &__x`)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (`const key_type &__x`) const
- `_Base_iterator erase (_Base_const_iterator __position)`
- `size_type erase (const key_type &__x)`
- **iterator** `erase (const_iterator __first, const_iterator __last)`

- `iterator erase` (`const_iterator` \_\_position)
- `node_type extract` (`const key_type` &\_\_key)
- `node_type extract` (`const_iterator` \_\_position)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator find` (`const _Kt` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator find` (`const _Kt` &\_\_x) `const`
- `iterator find` (`const key_type` &\_\_x)
- `const_iterator find` (`const key_type` &\_\_x) `const`
- `template<typename _InputIterator>`  
`void insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert` (`_Pair` &&\_\_x)
- `iterator insert` (`const value_type` &\_\_x)
- `iterator insert` (`const_iterator` \_\_hint, `node_type` &&\_\_nh)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert` (`const_iterator` \_\_position, `_Pair` &&\_\_x)
- `iterator insert` (`const_iterator` \_\_position, `const value_type` &\_\_x)
- `iterator insert` (`const_iterator` \_\_position, `value_type` &&\_\_x)
- `iterator insert` (`node_type` &&\_\_nh)
- `void insert` (`std::initializer_list< value_type >` \_\_list)
- `iterator insert` (`value_type` &&\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator lower_bound` (`const _Kt` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator lower_bound` (`const _Kt` &\_\_x) `const`
- `iterator lower_bound` (`const key_type` &\_\_x)
- `const_iterator lower_bound` (`const key_type` &\_\_x) `const`
- `multimap` & `operator=` (`const multimap` &)=default
- `multimap` & `operator=` (`initializer_list< value_type >` \_\_l)
- `multimap` & `operator=` (`multimap` &&)=default
- `const_reverse_iterator rbegin` () `const noexcept`
- `reverse_iterator rbegin` () `noexcept`
- `const_reverse_iterator rend` () `const noexcept`
- `reverse_iterator rend` () `noexcept`
- `void swap` (`multimap` &\_\_x) `noexcept( /*conditional */ )`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator upper_bound` (`const _Kt` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator upper_bound` (`const _Kt` &\_\_x) `const`
- `iterator upper_bound` (`const key_type` &\_\_x)
- `const_iterator upper_bound` (`const key_type` &\_\_x) `const`

## Protected Member Functions

- `constexpr void _M_swap` (`const _Safe_container` &\_\_x) `const noexcept`

## Friends

- `template<typename _IT, typename _SeqT, typename _CatT>`  
`class ::__gnu_debug:: _Safe_iterator`

### 5.684.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator =
std::allocator<std::pair<const _Key, _Tp>>>
class std::_debug::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class std::multimap with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [multimap.h](#)

## 5.685 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

```
#include <map>
```

### Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_Rep\_type::const\_iterator **const\_iterator**
- typedef \_Alloc\_traits::const\_pointer **const\_pointer**
- typedef \_Alloc\_traits::const\_reference **const\_reference**
- typedef \_Rep\_type::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Alloc\_traits::pointer **pointer**
- typedef \_Alloc\_traits::reference **reference**
- typedef \_Rep\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Rep\_type::size\_type **size\_type**
- typedef [std::pair](#)< const \_Key, \_Tp > **value\_type**

### Public Member Functions

- [multimap](#) ()=default
- template<typename \_InputIterator>  
[multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator>  
[multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator>  
[multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- [multimap](#) (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- [multimap](#) (const allocator\_type &\_\_a)
- [multimap](#) (const [multimap](#) &)=default
- [multimap](#) (const [multimap](#) &\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a)
- [multimap](#) (initializer\_list< [value\\_type](#) > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- [multimap](#) (initializer\_list< [value\\_type](#) > \_\_l, const allocator\_type &\_\_a)
- [multimap](#) ([multimap](#) &&)=default
- [multimap](#) ([multimap](#) &&\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept([is\\_nothrow\\_copy\\_constructible](#)< \_Compare >::value && \_Alloc\_traits::\_S\_always\_equal())
- [~multimap](#) ()=default

- const\_iterator [begin](#) () const noexcept
  - iterator [begin](#) () noexcept
  - const\_iterator [cbegin](#) () const noexcept
  - const\_iterator [cend](#) () const noexcept
  - void [clear](#) () noexcept
  - const\_reverse\_iterator [crbegin](#) () const noexcept
  - const\_reverse\_iterator [crend](#) () const noexcept
  - template<typename... \_Args>  
iterator [emplace](#) (\_Args &&... \_\_args)
  - template<typename... \_Args>  
iterator [emplace\\_hint](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool [empty](#) () const noexcept
  - const\_iterator [end](#) () const noexcept
  - iterator [end](#) () noexcept
  - size\_type [erase](#) (const key\_type &\_\_x)
  - iterator [erase](#) (const\_iterator \_\_first, const\_iterator \_\_last)
  - allocator\_type [get\\_allocator](#) () const noexcept
  - template<typename \_InputIterator>  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void [insert](#) (initializer\_list< value\_type > \_\_l)
  - key\_compare [key\\_comp](#) () const
  - size\_type [max\\_size](#) () const noexcept
  - multimap & operator= (const multimap &)=default
  - multimap & operator= (initializer\_list< value\_type > \_\_l)
  - multimap & operator= (multimap &&)=default
  - const\_reverse\_iterator [rbegin](#) () const noexcept
  - reverse\_iterator [rbegin](#) () noexcept
  - const\_reverse\_iterator [rend](#) () const noexcept
  - reverse\_iterator [rend](#) () noexcept
  - size\_type [size](#) () const noexcept
  - void [swap](#) (multimap &\_\_x) noexcept(*/\*conditional \*/*)
  - value\_compare [value\\_comp](#) () const
- 
- iterator [insert](#) (const value\_type &\_\_x)
  - iterator [insert](#) (value\_type &&\_\_x)
  - template<typename \_Pair>  
\_\_enable\_if\_t< [is\\_constructible](#)< value\_type, \_Pair >::value, iterator > [insert](#) (\_Pair &&\_\_x)
- 
- iterator [insert](#) (const\_iterator \_\_position, const value\_type &\_\_x)
  - iterator [insert](#) (const\_iterator \_\_position, value\_type &&\_\_x)
  - template<typename \_Pair>  
\_\_enable\_if\_t< [is\\_constructible](#)< value\_type, \_Pair && >::value, iterator > [insert](#) (const\_iterator \_\_position, \_Pair &&\_\_x)
- 
- iterator [erase](#) (const\_iterator \_\_position)
  - \_GLIBCXX\_ABI\_TAG\_CXX11 iterator [erase](#) (iterator \_\_position)

- iterator [find](#) (const key\_type &\_\_x)
- template<typename \_Kt>  
auto [find](#) (const \_Kt &\_\_x) -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))
- const\_iterator [find](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [find](#) (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))
- size\_type [count](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [count](#) (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_count\_tr(\_\_x))
- bool [contains](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [contains](#) (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x), void(), true)
- iterator [lower\\_bound](#) (const key\_type &\_\_x)
- template<typename \_Kt>  
auto [lower\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- const\_iterator [lower\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [lower\\_bound](#) (const \_Kt &\_\_x) const -> decltype(const\_iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- iterator [upper\\_bound](#) (const key\_type &\_\_x)
- template<typename \_Kt>  
auto [upper\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
- const\_iterator [upper\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [upper\\_bound](#) (const \_Kt &\_\_x) const -> decltype(const\_iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
- std::pair< iterator, iterator > [equal\\_range](#) (const key\_type &\_\_x)
- template<typename \_Kt>  
auto [equal\\_range](#) (const \_Kt &\_\_x) -> decltype(pair< iterator, iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))
- std::pair< const\_iterator, const\_iterator > [equal\\_range](#) (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto [equal\\_range](#) (const \_Kt &\_\_x) const -> decltype(pair< const\_iterator, const\_iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))



## Friends

- `template<typename _K1, typename _T1, typename _C1, typename _A1>`  
`bool operator< (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)`
- `template<typename _K1, typename _T1, typename _C1, typename _A1>`  
`bool operator== (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)`

### 5.685.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>>`  
`class std::multimap< _Key, _Tp, _Compare, _Alloc >`

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Since

C++98

#### Template Parameters

|                       |                                                                                         |
|-----------------------|-----------------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                                    |
| <code>_Tp</code>      | Type of mapped objects.                                                                 |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .            |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key,T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key,T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

### 5.685.2 Constructor & Destructor Documentation

#### `multimap()` [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap () [default]
```

Default constructor creates no elements.

#### `multimap()` [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    const _Compare & __comp,
    const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a `multimap` with no elements.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

**multimap()** [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
    const multimap<_Key, _Tp, _Compare, _Alloc > & ) [default]
```

Multimap copy constructor.

Whether the allocator is copied depends on the allocator traits.

**multimap()** [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
    multimap<_Key, _Tp, _Compare, _Alloc > && ) [default]
```

Multimap move constructor.

The newly-created multimap contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multimap.

**multimap()** [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
    initializer_list< value_type > __l,
    const _Compare & __comp = _Compare(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a multimap from an initializer\_list.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__l</code>    | An initializer_list.  |
| <code>__comp</code> | A comparison functor. |
| <code>__a</code>    | An allocator object.  |

Create a multimap consisting of copies of the elements from the initializer\_list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

**multimap()** [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
    const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

**multimap()** [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
    const multimap<_Key, _Tp, _Compare, _Alloc > & __m,
    const __type_identity_t< allocator_type > & __a) [inline]
```

Allocator-extended copy constructor.

**multimap()** [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    multimap< _Key, _Tp, _Compare, _Alloc > && __m,
    const __type_identity_t< allocator_type > & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

**multimap()** [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    initializer_list< value_type > __l,
    const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

**multimap()** [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

**multimap()** [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Builds a multimap from a range.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

**multimap()** [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    _InputIterator __first,
    _InputIterator __last,
```

```
const _Compare & __comp,
const allocator_type & __a = allocator_type()) [inline]
```

Builds a multimap from a range.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

#### ~multimap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
std::multimap<_Key, _Tp, _Compare, _Alloc>::~~multimap () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### 5.685.3 Member Function Documentation

#### begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

#### begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

#### cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

#### cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

**clear()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
void std::multimap< _Key, _Tp, _Compare, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains() [1/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
template<typename _Kt>
```

```
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::contains (
    const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                                                                                   |                                          |
|-----------------------------------------------------------------------------------|------------------------------------------|
|  | Key of (key, value) pairs to be located. |
|  |                                          |

**Returns**

True if there is any element with the specified key.

**contains() [2/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
bool std::multimap< _Key, _Tp, _Compare, _Alloc >::contains (
    const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                                                                                     |                                          |
|-------------------------------------------------------------------------------------|------------------------------------------|
|  | Key of (key, value) pairs to be located. |
|  |                                          |

**Returns**

True if there is any element with the specified key.

**count() [1/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
template<typename _Kt>
```

```
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::count (
    const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x)) [inline]
```

Finds the number of elements with given key.

**Parameters**

|                 |                                          |
|-----------------|------------------------------------------|
| <code>_↔</code> | Key of (key, value) pairs to be located. |
| <code>_X</code> |                                          |

**Returns**

Number of elements with specified key.

**count()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::count (
    const key_type & __x) const [inline]
```

Finds the number of elements with given key.

**Parameters**

|                 |                                          |
|-----------------|------------------------------------------|
| <code>_↔</code> | Key of (key, value) pairs to be located. |
| <code>_X</code> |                                          |

**Returns**

Number of elements with specified key.

**crbegin()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

**crend()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

**emplace()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace (
    _Args &&... __args) [inline]
```

Build and insert a std::pair into the multimap.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

**emplace\_hint()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args) [inline]
```

Builds and inserts a `std::pair` into the multimap.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                             |
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**empty()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::multimap<_Key, _Tp, _Compare, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the multimap is empty.

**end()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

**end()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

**equal\_range()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
template<typename _Kt>
```

```
auto std::multimap<_Key, _Tp, _Compare, _Alloc>::equal_range (
    const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
```

```
[inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

**equal\_range()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
template<typename _Kt>
```

```
auto std::multimap<_Key, _Tp, _Compare, _Alloc>::equal_range (
    const _Kt & __x) const -> decltype(pair<const_iterator, const_iterator>(_M_t._M_↵
```

```
equal_range_tr(__x))) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).



**equal\_range()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, iterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
    const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

**equal\_range()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< const_iterator, const_iterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
    const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

**erase()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
    const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                 |                              |
|-----------------|------------------------------|
| <code>_↔</code> | Key of element to be erased. |
| <code>_X</code> |                              |

**Returns**

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase() [2/4]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::erase (
    const_iterator __first,
    const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a multimap.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased .  |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase() [3/4]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::erase (
    const_iterator __position) [inline]
```

Erases an element from a multimap.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
    iterator __position) [inline]
```

Erases an element from a multimap.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
    const _Kt & __x) -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a multimap.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**find()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
    const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a multimap.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
    const key_type & __x) [inline]
```

Tries to locate an element in a multimap.

**Parameters**

|                                                                                                   |                                         |
|---------------------------------------------------------------------------------------------------|-----------------------------------------|
|  <code>_x</code> | Key of (key, value) pair to be located. |
|---------------------------------------------------------------------------------------------------|-----------------------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
    const key_type & __x) const [inline]
```

Tries to locate an element in a multimap.

**Parameters**

|                                                                                                     |                                         |
|-----------------------------------------------------------------------------------------------------|-----------------------------------------|
|  <code>_x</code> | Key of (key, value) pair to be located. |
|-----------------------------------------------------------------------------------------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( end() ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::multimap< _Key, _Tp, _Compare, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Get a copy of the memory allocation object.

**insert()** [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert()** [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, iterator > std::multimap< _Key, ↵
_Tp, _Compare, _Alloc >::insert (
    _Pair && __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|                                    |                                                                                   |
|------------------------------------|-----------------------------------------------------------------------------------|
| <code>↵</code><br><code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------------------------|-----------------------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

**insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
    const value_type & __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|                                    |                                                                                   |
|------------------------------------|-----------------------------------------------------------------------------------|
| <code>↵</code><br><code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------------------------|-----------------------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Referenced by `std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert()`.

**insert()** [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::multimap< _Key,
_Tp, _Compare, _Alloc >::insert (
    const_iterator __position,
    _Pair && __x) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>        | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
    const_iterator __position,
    const value_type & __x) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>        | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
    const_iterator __position,
    value_type && __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
    initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of `std::pairs` into the multimap.

**Parameters**

|                  |                                                                                 |
|------------------|---------------------------------------------------------------------------------|
| <code>__l</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of pairs to be inserted. |
|------------------|---------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

**insert()** [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::insert (
    value_type && __x) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

**key\_comp()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
key_compare std::multimap<_Key, _Tp, _Compare, _Alloc>::key_comp () const [inline]
```

Returns the key comparison object out of which the multimap was constructed.

**lower\_bound()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::multimap<_Key, _Tp, _Compare, _Alloc>::lower_bound (
    const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
```



```
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const _Kt & __x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

#### Parameters

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↵</code> | Key of (key, value) pair to be located. |
| <code>_X</code> |                                         |

#### Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

#### **lower\_bound()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

#### Parameters

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↵</code> | Key of (key, value) pair to be located. |
| <code>_X</code> |                                         |

#### Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

#### **lower\_bound()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

#### Parameters

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↵</code> | Key of (key, value) pair to be located. |
| <code>_X</code> |                                         |

#### Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

**max\_size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap<_Key, _Tp, _Compare, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the multimap.

**operator=()** [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap<_Key, _Tp, _Compare, _Alloc >::operator= (
    const multimap<_Key, _Tp, _Compare, _Alloc > & ) [default]
```

Multimap assignment operator.

Whether the allocator is copied depends on the allocator traits.

**operator=()** [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap<_Key, _Tp, _Compare, _Alloc >::operator= (
    initializer_list<value_type > __l) [inline]
```

Multimap list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills a multimap with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap<_Key, _Tp, _Compare, _Alloc >::operator= (
    multimap<_Key, _Tp, _Compare, _Alloc > && ) [default]
```

Move assignment operator.

**rbegin()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::rbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

**rbegin()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

**rend()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

**rend()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

**size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the multimap.

**swap()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
void std::multimap< _Key, _Tp, _Compare, _Alloc >::swap (
    multimap< _Key, _Tp, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another multimap.

**Parameters**

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__x</code> | A multimap of the same element and allocator types. |
|------------------|-----------------------------------------------------|

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Referenced by [std::swap\(\)](#).

**upper\_bound()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
template<typename _Kt>
auto std::multimap<_Key, _Tp, _Compare, _Alloc>::upper_bound (
    const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))    [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Iterator pointing to the first element greater than key, or end().

#### upper\_bound() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::multimap<_Key, _Tp, _Compare, _Alloc>::upper_bound (
    const _Kt & __x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

#### upper\_bound() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::upper_bound (
    const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
    const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to first iterator greater than key, or end().

**value\_comp()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::value_comp () const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

The documentation for this class was generated from the following files:

- [stl\\_map.h](#)
- [stl\\_multimap.h](#)

**5.686 std::multiplies<\_Tp> Struct Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for `std::multiplies<_Tp>`:

**Public Types**

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

**Public Member Functions**

- constexpr **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

**5.686.1 Detailed Description**

```
template<typename _Tp>
struct std::multiplies< _Tp >
```

One of the [math functors](#).

**5.686.2 Member Typedef Documentation****first\_argument\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

**result\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

**5.687 std::multiplies< void > Struct Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::multiplies< void >:

**Public Types**

- typedef void [first\\_argument\\_type](#)
- typedef \_\_is\_transparent [is\\_transparent](#)

- typedef void [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t) \*[std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t) \*[std::forward](#)< \_Up >(\_\_u))
- constexpr void **operator()** (const void &\_\_x, const void &\_\_y) const

### 5.687.1 Detailed Description

One of the [math functors](#).

### 5.687.2 Member Typedef Documentation

#### first\_argument\_type

typedef void [std::binary\\_function](#)< void, void, void >::first\_argument\_type  
first\_argument\_type is the type of the first argument

#### result\_type

typedef void [std::binary\\_function](#)< void, void, void >::result\_type  
result\_type is the return type

#### second\_argument\_type

typedef void [std::binary\\_function](#)< void, void, void >::second\_argument\_type  
second\_argument\_type is the type of the second argument  
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.688 std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator > Class Template Reference

```
#include <multiset.h>
```

Inheritance diagram for std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator >:



### Public Types

- typedef \_Allocator **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< [\\_Base\\_const\\_iterator](#), [multiset](#) > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**

- typedef \_Base::const\_reference **const\_reference**
- typedef std::reverse\_iterator< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_\_gnu\_debug::\_Safe\_iterator< \_Base\_iterator, multiset > **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- using **node\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef std::reverse\_iterator< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Key **value\_type**

### Public Member Functions

- **multiset** (\_Base\_ref \_\_x)
- template<typename \_InputIterator>  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator>  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- **multiset** (const \_Compare &\_\_comp, const \_Allocator &\_\_a=\_Allocator())
- **multiset** (const allocator\_type &\_\_a)
- **multiset** (const multiset &)=default
- **multiset** (const multiset &\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a)
- **multiset** (initializer\_list< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a←a=allocator\_type())
- **multiset** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a)
- **multiset** (multiset &&)=default
- **multiset** (multiset &&\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept(noexcept(\_Base(std::move(←\_m), \_\_a)))
- \_\_attribute\_\_((abi\_tag("cxx11"))) iterator erase(const\_iterator \_\_first
- \_\_attribute\_\_((abi\_tag("cxx11"))) iterator erase(const\_iterator \_\_position)
- const \_Base & \_M\_base () const noexcept
- \_Base & \_M\_base () noexcept
- const\_iterator begin () const noexcept
- iterator begin () noexcept
- const\_iterator cbegin () const noexcept
- const\_iterator cend () const noexcept
- void clear () noexcept
- const\_reverse\_iterator crbegin () const noexcept
- const\_reverse\_iterator crend () const noexcept
- template<typename... \_Args>  
iterator emplace (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator emplace\_hint (const\_iterator \_\_pos, \_Args &&... \_\_args)
- const\_iterator end () const noexcept
- iterator end () noexcept
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent< \_Compare, \_Kt>::type>  
std::pair< iterator, iterator > equal\_range (const \_Kt &\_\_x)



- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>  
std::pair< const_iterator, const_iterator > equal_range (const _Kt &__x) const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `_Base_iterator erase (_Base_const_iterator __position)`
- `size_type erase (const key_type &__x)`
- `node_type extract (const key_type &__key)`
- `node_type extract (const_iterator __position)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>  
iterator find (const _Kt &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>  
const_iterator find (const _Kt &__x) const`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `for (_Base_const_iterator __victim=__first.base(); __victim != __last.base(); ++__victim)`
- `template<typename _InputIterator>  
void insert (_InputIterator __first, _InputIterator __last)`
- `iterator insert (const value_type &__x)`
- `iterator insert (const_iterator __hint, node_type &&__nh)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type &&__x)`
- `void insert (initializer_list< value_type > __l)`
- `iterator insert (node_type &&__nh)`
- `iterator insert (value_type &&__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>  
iterator lower_bound (const _Kt &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>  
const_iterator lower_bound (const _Kt &__x) const`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `multiset & operator= (const multiset &)=default`
- `multiset & operator= (initializer_list< value_type > __l)`
- `multiset & operator= (multiset &&)=default`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `reverse_iterator rend () noexcept`
- `void swap (multiset &__x) noexcept(/*conditional */)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>  
iterator upper_bound (const _Kt &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>  
const_iterator upper_bound (const _Kt &__x) const`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `while (false)`

## Public Attributes

- `const_iterator __last`
- `do`
- `return`

**Protected Member Functions**

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

**Friends**

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug::\_\_Safe\_iterator

**5.688.1 Detailed Description**

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Allocator = std::allocator<\_Key>>

class std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator >

Class std::multiset with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [multiset.h](#)

**5.689 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference**

```
#include <set>
```

**Public Types**

- typedef \_Alloc **allocator\_type**
- typedef \_Rep\_type::const\_iterator **const\_iterator**
- typedef \_Alloc\_traits::const\_pointer **const\_pointer**
- typedef \_Alloc\_traits::const\_reference **const\_reference**
- typedef \_Rep\_type::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::const\_iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Alloc\_traits::pointer **pointer**
- typedef \_Alloc\_traits::reference **reference**
- typedef \_Rep\_type::const\_reverse\_iterator **reverse\_iterator**
- typedef \_Rep\_type::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Key **value\_type**

**Public Member Functions**

- [multiset](#) ()=default
- template<typename \_InputIterator>  
[multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator>  
[multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator>  
[multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- [multiset](#) (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- [multiset](#) (const allocator\_type &\_\_a)

- `multiset` (const `multiset` &)=default
  - `multiset` (const `multiset` &\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a)
  - `multiset` (`initializer_list`< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a←a=allocator\_type())
  - `multiset` (`initializer_list`< value\_type > \_\_l, const allocator\_type &\_\_a)
  - `multiset` (`multiset` &&)=default
  - `multiset` (`multiset` &&\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept(is\_nothrow\_copy\_constructible< \_Compare >::value && \_Alloc\_traits::\_S\_always\_equal())
  - `~multiset` ()=default
  - iterator `begin` () const noexcept
  - iterator `cbegin` () const noexcept
  - iterator `cend` () const noexcept
  - void `clear` () noexcept
  - `reverse_iterator` `crbegin` () const noexcept
  - `reverse_iterator` `crend` () const noexcept
  - template<typename... \_Args>  
iterator `emplace` (\_Args &&... \_\_args)
  - template<typename... \_Args>  
iterator `emplace_hint` (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool `empty` () const noexcept
  - iterator `end` () const noexcept
  - size\_type `erase` (const key\_type &\_\_x)
  - \_GLIBCXX\_ABI\_TAG\_CXX11 iterator `erase` (const\_iterator \_\_first, const\_iterator \_\_last)
  - \_GLIBCXX\_ABI\_TAG\_CXX11 iterator `erase` (const\_iterator \_\_position)
  - allocator\_type `get_allocator` () const noexcept
  - template<typename \_InputIterator>  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - iterator `insert` (const value\_type &\_\_x)
  - iterator `insert` (const\_iterator \_\_position, const value\_type &\_\_x)
  - iterator `insert` (const\_iterator \_\_position, value\_type &&\_\_x)
  - void `insert` (`initializer_list`< value\_type > \_\_l)
  - iterator `insert` (value\_type &&\_\_x)
  - key\_compare `key_comp` () const
  - size\_type `max_size` () const noexcept
  - `multiset` & `operator=` (const `multiset` &)=default
  - `multiset` & `operator=` (`initializer_list`< value\_type > \_\_l)
  - `multiset` & `operator=` (`multiset` &&)=default
  - `reverse_iterator` `rbegin` () const noexcept
  - `reverse_iterator` `rend` () const noexcept
  - size\_type `size` () const noexcept
  - void `swap` (`multiset` &\_\_x) noexcept(\*conditional \*)
  - value\_compare `value_comp` () const
- 
- size\_type `count` (const key\_type &\_\_x) const
  - template<typename \_Kt>  
auto `count` (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_count\_tr(\_\_x))
- 
- bool `contains` (const key\_type &\_\_x) const

- template<typename \_Kt>  
auto **contains** (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x), void(), true)
- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto **find** (const \_Kt &\_\_x) -> decltype(iterator{ \_M\_t.\_M\_find\_tr(\_\_x) })
- template<typename \_Kt>  
auto **find** (const \_Kt &\_\_x) const -> decltype(const\_iterator{ \_M\_t.\_M\_find\_tr(\_\_x) })
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto **lower\_bound** (const \_Kt &\_\_x) -> decltype(iterator{ \_M\_t.\_M\_lower\_bound\_tr(\_\_x) })
- template<typename \_Kt>  
auto **lower\_bound** (const \_Kt &\_\_x) const -> decltype(iterator{ \_M\_t.\_M\_lower\_bound\_tr(\_\_x) })
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto **upper\_bound** (const \_Kt &\_\_x) -> decltype(iterator{ \_M\_t.\_M\_upper\_bound\_tr(\_\_x) })
- template<typename \_Kt>  
auto **upper\_bound** (const \_Kt &\_\_x) const -> decltype(iterator{ \_M\_t.\_M\_upper\_bound\_tr(\_\_x) })
- std::pair< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- std::pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto **equal\_range** (const \_Kt &\_\_x) -> decltype(pair< iterator, iterator >{ \_M\_t.\_M\_equal\_range\_tr(\_\_x) })
- template<typename \_Kt>  
auto **equal\_range** (const \_Kt &\_\_x) const -> decltype(pair< iterator, iterator >{ \_M\_t.\_M\_equal\_range\_tr(\_\_x) })

## Friends

- template<typename \_K1, typename \_C1, typename \_A1>  
bool **operator**< (const multiset< \_K1, \_C1, \_A1 > &, const multiset< \_K1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_C1, typename \_A1>  
bool **operator**== (const multiset< \_K1, \_C1, \_A1 > &, const multiset< \_K1, \_C1, \_A1 > &)

### 5.689.1 Detailed Description

**template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Alloc = std::allocator<\_Key>>**  
**class std::multiset< \_Key, \_Compare, \_Alloc >**

A standard container made up of elements, which can be retrieved in logarithmic time.

Since

C++98

### Template Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                         |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> . |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .             |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

### 5.689.2 Constructor & Destructor Documentation

#### `multiset()` [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::multiset< _Key, _Compare, _Alloc >::multiset () [default]
```

Default constructor creates no elements.

#### `multiset()` [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::multiset< _Key, _Compare, _Alloc >::multiset (
    const _Compare & __comp,
    const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a multiset with no elements.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | Comparator to use.   |
| <code>__a</code>    | An allocator object. |

#### `multiset()` [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _InputIterator>
```

```
std::multiset< _Key, _Compare, _Alloc >::multiset (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Builds a multiset from a range.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a multiset consisting of copies of the elements from `[first,last)`. This is linear in `N` if the range is already sorted, and `NlogN` otherwise (where `N` is `distance(__first,__last)`).

**multiset()** [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    _InputIterator __first,
    _InputIterator __last,
    const _Compare & __comp,
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a multiset from a range.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a multiset consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

**multiset()** [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    const multiset< _Key, _Compare, _Alloc > & ) [default]
```

Multiset copy constructor.

Whether the allocator is copied depends on the allocator traits.

**multiset()** [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    multiset< _Key, _Compare, _Alloc > && ) [default]
```

Multiset move constructor.

The newly-created multiset contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multiset.

**multiset()** [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    initializer_list< value_type > __l,
    const _Compare & __comp = _Compare(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a multiset from an `initializer_list`.

**Parameters**

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> . |
|------------------|------------------------------------|

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__comp</code> | A comparison functor. |
| <code>__a</code>    | An allocator object.  |

Create a multiset consisting of copies of the elements from the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

**multiset() [8/12]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc>::multiset (
    const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

**multiset() [9/12]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc>::multiset (
    const multiset<_Key, _Compare, _Alloc> & __m,
    const __type_identity_t< allocator_type > & __a) [inline]
```

Allocator-extended copy constructor.

**multiset() [10/12]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc>::multiset (
    multiset<_Key, _Compare, _Alloc> && __m,
    const __type_identity_t< allocator_type > & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

**multiset() [11/12]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc>::multiset (
    initializer_list< value_type > __l,
    const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

**multiset() [12/12]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator>
std::multiset<_Key, _Compare, _Alloc>::multiset (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

**~multiset()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
std::multiset<_Key, _Compare, _Alloc >::~~multiset () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**5.689.3 Member Function Documentation****begin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::multiset<_Key, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

**cbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::multiset<_Key, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

**cend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::multiset<_Key, _Compare, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

**clear()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
void std::multiset<_Key, _Compare, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains() [1/2]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
template<typename _Kt>
```

```
auto std::multiset<_Key, _Compare, _Alloc >::contains (
    const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>↵</code>   | Key of elements to be located. |
| <code>__x</code> |                                |



**Returns**

True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
bool std::multiset< _Key, _Compare, _Alloc >::contains (
    const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                                    |                                |
|------------------------------------|--------------------------------|
| <code>↵</code><br><code>__x</code> | Key of elements to be located. |
|------------------------------------|--------------------------------|

**Returns**

True if there is any element with the specified key.

**count()** [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::count (
    const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x)) [inline]
```

Finds the number of elements with given key.

**Parameters**

|                                    |                                |
|------------------------------------|--------------------------------|
| <code>↵</code><br><code>__x</code> | Key of elements to be located. |
|------------------------------------|--------------------------------|

**Returns**

Number of elements with specified key.

**count()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::count (
    const key_type & __x) const [inline]
```

Finds the number of elements with given key.

**Parameters**

|                                    |                                |
|------------------------------------|--------------------------------|
| <code>↵</code><br><code>__x</code> | Key of elements to be located. |
|------------------------------------|--------------------------------|

**Returns**

Number of elements with specified key.

**crbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

**crend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

**emplace()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename... _Args>
```

```
iterator std::multiset< _Key, _Compare, _Alloc >::emplace (
    _Args &&... __args) [inline]
```

Builds and inserts an element into the multiset.

**Parameters**

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate the element instance to be inserted. |
|---------------------|-----------------------------------------------------------------|

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

**emplace\_hint()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename... _Args>
```

```
iterator std::multiset< _Key, _Compare, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args) [inline]
```

Builds and inserts an element into the multiset.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element instance to be inserted.               |

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

**empty()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
bool std::multiset< _Key, _Compare, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the set is empty.

**end()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
iterator std::multiset< _Key, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

**equal\_range()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _Kt>
```

```
auto std::multiset< _Key, _Compare, _Alloc >::equal_range (
    const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
```

```
[inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**equal\_range()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::equal_range (
    const _Kt & __x) const -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_↔
tr(__x))) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>_↔</code>  | Key to be located. |
| <code>__x</code> |                    |

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

#### equal\_range() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
std::pair< iterator, iterator > std::multiset< _Key, _Compare, _Alloc >::equal_range (
    const key_type & __x) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>_↔</code>  | Key to be located. |
| <code>__x</code> |                    |

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

#### equal\_range() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
std::pair< const_iterator, const_iterator > std::multiset< _Key, _Compare, _Alloc >::equal_range
(
    const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↔</code> | Key to be located. |
| <code>_x</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**erase() [1/3]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔  
_Key>>  
size_type std::multiset< _Key, _Compare, _Alloc >::erase (  
    const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                 |                              |
|-----------------|------------------------------|
| <code>_↔</code> | Key of element to be erased. |
| <code>_x</code> |                              |

**Returns**

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase() [2/3]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔  
_Key>>  
_GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase (  
    const_iterator __first,  
    const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a multiset.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase (
    const_iterator __position) [inline]
```

Erases an element from a multiset.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::find (
    const _Kt & __x) -> decltype(iterator{_M_t._M_find_tr(__x)}) [inline]
```

Tries to locate an element in a set.

**Parameters**

|                        |                        |
|------------------------|------------------------|
| <code>↵<br/>__x</code> | Element to be located. |
|------------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**find()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::find (
    const _Kt & __x) const -> decltype(const_iterator{_M_t._M_find_tr(__x)}) [inline]
```

Tries to locate an element in a set.

**Parameters**

|                        |                        |
|------------------------|------------------------|
| <code>↵<br/>__x</code> | Element to be located. |
|------------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::find (
    const key_type & __x) [inline]
```

Tries to locate an element in a set.

**Parameters**

|                                    |                        |
|------------------------------------|------------------------|
| <code>↵</code><br><code>__x</code> | Element to be located. |
|------------------------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::find (
    const key_type & __x) const [inline]
```

Tries to locate an element in a set.

**Parameters**

|                                    |                        |
|------------------------------------|------------------------|
| <code>↵</code><br><code>__x</code> | Element to be located. |
|------------------------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
allocator_type std::multiset< _Key, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]
```

Returns the memory allocation object.

**insert()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _InputIterator>
void std::multiset< _Key, _Compare, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

A template function that tries to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
    const value_type & __x) [inline]
```

Inserts an element into the multiset.

**Parameters**

|                                    |                         |
|------------------------------------|-------------------------|
| <code>↵</code><br><code>__x</code> | Element to be inserted. |
|------------------------------------|-------------------------|

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Referenced by [std::multiset< \\_Key, \\_Cmp, polymorphic\\_allocator< \\_Key > >::insert\(\)](#).

**insert()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
    const_iterator __position,
    const value_type & __x) [inline]
```

Inserts an element into the multiset.

**Parameters**

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>        | Element to be inserted.                                                       |



**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::multiset< _Key, _Compare, _Alloc >::insert (
    initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the multiset.

**Parameters**

|   |                                                                                    |
|---|------------------------------------------------------------------------------------|
| ↵ | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
| ↵ |                                                                                    |
| ↵ |                                                                                    |
| ↵ |                                                                                    |
| / |                                                                                    |

Complexity similar to that of the range constructor.

**key\_comp()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
key_compare std::multiset< _Key, _Compare, _Alloc >::key_comp () const [inline]
```

Returns the comparison object.

**lower\_bound()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::lower_bound (
    const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|     |                    |
|-----|--------------------|
| ↵   | Key to be located. |
| __x |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

**lower\_bound()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::multiset<_Key, _Compare, _Alloc>::lower_bound (
    const _Kt & __x) const -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))    [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::multiset<_Key, _Compare, _Alloc>::lower_bound (
    const key_type & __x)    [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::multiset<_Key, _Compare, _Alloc>::lower_bound (
    const key_type & __x) const    [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**max\_size()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the set.

**operator=()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset & std::multiset< _Key, _Compare, _Alloc >::operator= (
    const multiset< _Key, _Compare, _Alloc > & ) [default]
```

Multiset assignment operator.

Whether the allocator is copied depends on the allocator traits.

**operator=()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset & std::multiset< _Key, _Compare, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

Multiset list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↵ | An initializer_list. |
| ↵ |                      |
| ↵ |                      |
| ↵ |                      |
| / |                      |

This function fills a multiset with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset & std::multiset< _Key, _Compare, _Alloc >::operator= (
    multiset< _Key, _Compare, _Alloc > && ) [default]
```

Move assignment operator.

**rbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

### rend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

### size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
size_type std::multiset< _Key, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the set.

### swap()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
void std::multiset< _Key, _Compare, _Alloc >::swap (
    multiset< _Key, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another multiset.

#### Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__x</code> | A multiset of the same element and allocator types. |
|------------------|-----------------------------------------------------|

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Referenced by [std::swap\(\)](#).

### upper\_bound() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _Kt>
```

```
auto std::multiset< _Key, _Compare, _Alloc >::upper_bound (
    const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x))) [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

**upper\_bound()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::upper_bound (
    const _Kt & __x) const -> decltype(iterator(_M.t._M_upper_bound_tr(__x)))    [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
    const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
    const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to the first element greater than key, or end().

**value\_comp()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
value_compare std::multiset< _Key, _Compare, _Alloc >::value_comp () const [inline]
```

Returns the comparison object.

The documentation for this class was generated from the following file:

- [stl\\_multiset.h](#)

**5.690 `__gnu_parallel::multiway_mergesort_exact_tag` Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::multiway_mergesort_exact_tag`:

**Public Member Functions**

- `multiway_mergesort_exact_tag` ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([\\_ThreadIndex](#) \_\_num\_threads)

**5.690.1 Detailed Description**

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

**5.690.2 Member Function Documentation****`__get_num_threads()`**

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

**Returns**

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

**set\_num\_threads()**

```
void __gnu_parallel::parallel_tag::set_num_threads (
    _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

**Parameters**

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

**5.691 \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::multiway_mergesort_sampling_tag`:

**Public Member Functions**

- **multiway\_mergesort\_sampling\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

**5.691.1 Detailed Description**

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

**5.691.2 Member Function Documentation****\_\_get\_num\_threads()**

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

**Returns**

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

**set\_num\_threads()**

```
void __gnu_parallel::parallel_tag::set_num_threads (
    _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

**Parameters**

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

**5.692 \_\_gnu\_parallel::multiway\_mergesort\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for \_\_gnu\_parallel::multiway\_mergesort\_tag:

**Public Member Functions**

- **multiway\_mergesort\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

**5.692.1 Detailed Description**

Forces parallel sorting using multiway mergesort at compile time.

**5.692.2 Member Function Documentation****\_\_get\_num\_threads()**

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

**Returns**

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).



**set\_num\_threads()**

```
void __gnu_parallel::parallel_tag::set_num_threads (
    __ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

**Parameters**

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

**5.693 std::mutex Class Reference**

```
#include <mutex>
```

**Public Types**

- typedef `__native_type * native_handle_type`

**Public Member Functions**

- **mutex** (const [mutex](#) &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** () noexcept
- [mutex](#) & **operator=** (const [mutex](#) &)=delete
- bool **try\_lock** () noexcept
- void **unlock** ()

**5.693.1 Detailed Description**

The standard mutex type.

A simple, non-recursive, non-timed mutex.

Do not call `lock()` and `unlock()` directly, use a scoped lock type such as `std::unique_lock`, `std::lock_guard`, or (since C++17) `std::scoped_lock`.

Since

C++11

The documentation for this class was generated from the following file:

- [std\\_mutex.h](#)

**5.694 std::negate<\_Tp> Struct Template Reference**

```
#include <std_function.h>
```

Inheritance diagram for `std::negate< _Tp >`:



### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `_Tp` [result\\_type](#)

### Public Member Functions

- constexpr `_Tp` **operator()** (const `_Tp` &\_\_x) const

#### 5.694.1 Detailed Description

```
template<typename _Tp>
struct std::negate< _Tp >
```

One of the [math functors](#).

#### 5.694.2 Member Typedef Documentation

##### **argument\_type**

```
typedef _Tp std::unary_function< _Tp, _Tp >::argument_type [inherited]
argument_type is the type of the argument
```

##### **result\_type**

```
typedef _Tp std::unary_function< _Tp, _Tp >::result_type [inherited]
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.695 `std::negate< void >` Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::negate< void >`:



### Public Types

- typedef void [argument\\_type](#)
- typedef `__is_transparent` [is\\_transparent](#)
- typedef void [result\\_type](#)

### Public Member Functions

- template<typename `_Tp`>  
constexpr auto **operator()** (`_Tp` &&\_\_t) const noexcept(noexcept([std::forward](#)< `_Tp` >(\_\_t))) -> decltype([std::forward](#)< `_Tp` >(\_\_t))
- constexpr void **operator()** (const void &\_\_x) const

#### 5.695.1 Detailed Description

One of the [math functors](#).

#### 5.695.2 Member Typedef Documentation

##### **argument\_type**

typedef void [std::unary\\_function](#)< void, void >::argument\_type  
argument\_type is the type of the argument

##### **result\_type**

typedef void [std::unary\\_function](#)< void, void >::result\_type  
result\_type is the return type

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.696 `std::negative_binomial_distribution< _IntType >` Class Template Reference

```
#include <random>
```

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_IntType [result\\_type](#)

## Public Member Functions

- **negative\_binomial\_distribution** (\_IntType \_\_k, double \_\_p=0.5)
- **negative\_binomial\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_IntType **k** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_IntType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::negative\\_binomial\\_distribution](#)< \_IntType1 > &\_\_x)
- bool **operator==** (const [negative\\_binomial\\_distribution](#) &\_\_d1, const [negative\\_binomial\\_distribution](#) &\_\_d2)
- template<typename \_IntType1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::negative\\_binomial\\_distribution](#)< \_IntType1 > &\_\_x)

### 5.696.1 Detailed Description

**template<typename \_IntType = int>**  
**class std::negative\_binomial\_distribution< \_IntType >**

A negative\_binomial\_distribution random number distribution.

The formula for the negative binomial probability mass function is  $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Since

C++11

### 5.696.2 Member Typedef Documentation

#### result\_type

```
template<typename _IntType = int>
```

```
typedef _IntType std::negative_binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

### 5.696.3 Member Function Documentation

#### k()

```
template<typename _IntType = int>
```

```
_IntType std::negative_binomial_distribution< _IntType >::k () const [inline]
```

Return the  $k$  parameter of the distribution.

#### max()

```
template<typename _IntType = int>
```

```
result_type std::negative_binomial_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _IntType = int>
```

```
result_type std::negative_binomial_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()

```
template<typename _IntType>
```

```
template<typename _UniformRandomNumberGenerator>
```

```
negative_binomial_distribution< _IntType >::result_type std::negative_binomial_distribution< _↵
```

```
_IntType >::operator() (
```

```
    _UniformRandomNumberGenerator & __urng)
```

Generating functions.

#### p()

```
template<typename _IntType = int>
```

```
double std::negative_binomial_distribution< _IntType >::p () const [inline]
```

Return the  $p$  parameter of the distribution.

#### param() [1/2]

```
template<typename _IntType = int>
```

```
param_type std::negative_binomial_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### param() [2/2]

```
template<typename _IntType = int>
```

```
void std::negative_binomial_distribution< _IntType >::param (
```

```
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::reset () [inline]
Resets the distribution state.
```

**5.696.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::negative_binomial_distribution< _IntType1 > & __x) [friend]
Inserts a negative_binomial_distribution random number distribution __x into the output stream __os.
```

## Parameters

|                   |                                                              |
|-------------------|--------------------------------------------------------------|
| <code>__os</code> | An output stream.                                            |
| <code>__x</code>  | A negative_binomial_distribution random number distribution. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _IntType = int>
bool operator== (
    const negative_binomial_distribution< _IntType > & __d1,
    const negative_binomial_distribution< _IntType > & __d2) [friend]
```

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::negative_binomial_distribution< _IntType1 > & __x) [friend]
Extracts a negative_binomial_distribution random number distribution __x from the input stream __is.
```

## Parameters

|                   |                  |
|-------------------|------------------|
| <code>__is</code> | An input stream. |
|-------------------|------------------|

|                                                                                   |                                                                  |
|-----------------------------------------------------------------------------------|------------------------------------------------------------------|
|  | A negative_binomial_distribution random number generator engine. |
|-----------------------------------------------------------------------------------|------------------------------------------------------------------|

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.697 std::nested\_exception Class Reference

```
#include <exception>
```

### Public Member Functions

- [nested\\_exception](#) () noexcept
- [nested\\_exception](#) (const [nested\\_exception](#) &) noexcept=default
- [exception\\_ptr nested\\_ptr](#) () const noexcept
- [nested\\_exception](#) & [operator=](#) (const [nested\\_exception](#) &) noexcept=default
- void [rethrow\\_nested](#) () const

#### 5.697.1 Detailed Description

Mixin class that stores the current exception.

This type can be used via `std::throw_with_nested` to store the current exception nested within another exception.

Since

C++11

See also

`std::throw_with_nested`

#### 5.697.2 Constructor & Destructor Documentation

##### `nested_exception()`

```
std::nested_exception::nested_exception () [inline], [noexcept]
```

The default constructor stores the current exception (if any).

References [std::current\\_exception\(\)](#).

#### 5.697.3 Member Function Documentation

##### `nested_ptr()`

```
exception\_ptr std::nested_exception::nested_ptr () const [inline], [noexcept]
```

Access the stored exception.

**rethrow\_nested()**

```
void std::nested_exception::rethrow_nested () const [inline]
```

Rethrow the stored exception, or terminate if none was stored.

References [std::rethrow\\_exception\(\)](#), and [std::terminate\(\)](#).

The documentation for this class was generated from the following file:

- [nested\\_exception.h](#)

**5.698 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference**

```
#include <throw_allocator.h>
```

**5.698.1 Detailed Description**

Never enter the condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

**5.699 `__gnu_cxx::random_condition::never_adjustor` Struct Reference**

```
#include <throw_allocator.h>
```

**5.699.1 Detailed Description**

Never enter the condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

**5.700 `__gnu_cxx::new_allocator<_Tp>` Class Template Reference**

```
#include <ext/new_allocator.h>
```

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:





## Public Types

- typedef std::ptrdiff\_t **difference\_type**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **new\_allocator** (const [new\\_allocator](#) &) noexcept
- template<typename \_Tp1>  
  **new\_allocator** (const [new\\_allocator](#)<\_Tp1> &) noexcept
- \_Tp \* **allocate** (size\_type \_\_n, const void \* \_\_p) noexcept
- void **deallocate** (\_Tp \* \_\_p, size\_type \_\_n)

### 5.700.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::new_allocator<_Tp>
```

An allocator that uses global `new`, as per C++03 [20.4.1].  
This is precisely the allocator defined in the C++ Standard.

- all allocation calls `operator new`
- all deallocation calls `operator delete`

This is a non-standard extension that can be used to guarantee allocation from `new` even if the library has been configured to use a different implementation for `std::allocator`.

#### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

The documentation for this class was generated from the following file:

- [ext/new\\_allocator.h](#)

### 5.701 \_\_gnu\_pbds::detail::no\_throw\_copies< Key, Mapped > Struct Template Reference

```
#include <types_traits.hpp>
```

## Public Types

- typedef integral\_constant< int, \_\_simple > **indicator**

## Static Public Attributes

- static const bool **\_\_simple**

### 5.701.1 Detailed Description

```
template<typename Key, typename Mapped>
struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
```

Primary template.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

**5.702 `__gnu_pbds::detail::no_throw_copies< Key, null_type >` Struct Template Reference**

```
#include <types_traits.hpp>
```

**Public Types**

- typedef integral\_constant< int, \_\_simple > **indicator**
- typedef integral\_constant< int, is\_simple< Key >::value > **indicator**

**Static Public Attributes**

- static const bool **\_\_simple**

**5.702.1 Detailed Description**

```
template<typename Key>
struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
```

Specialization.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

**5.703 `std::normal_distribution< _RealType >` Class Template Reference**

```
#include <random>
```

**Classes**

- struct [param\\_type](#)

**Public Types**

- typedef \_RealType [result\\_type](#)

**Public Member Functions**

- **normal\_distribution** (const [param\\_type](#) &\_\_p)
- **normal\_distribution** ([result\\_type](#) \_\_mean, [result\\_type](#) \_\_stddev=[result\\_type](#)(1))
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>
 void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>
 void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>
 void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- \_RealType **mean** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>
 [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>
 [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const

- void `param` (const `param_type` &\_\_param)
- void `reset` ()
- `_RealType stddev` () const

## Friends

- template<typename `_RealType1`, typename `_CharT`, typename `_Traits`>  
`std::basic_ostream`< `_CharT`, `_Traits` > & `operator<<` (`std::basic_ostream`< `_CharT`, `_Traits` > &\_\_os, const `std::normal_distribution`< `_RealType1` > &\_\_x)
- template<typename `_RealType1`>  
bool `operator==` (const `std::normal_distribution`< `_RealType1` > &\_\_d1, const `std::normal_distribution`< `_RealType1` > &\_\_d2)
- template<typename `_RealType1`, typename `_CharT`, typename `_Traits`>  
`std::basic_istream`< `_CharT`, `_Traits` > & `operator>>` (`std::basic_istream`< `_CharT`, `_Traits` > &\_\_is, `std::normal_distribution`< `_RealType1` > &\_\_x)

### 5.703.1 Detailed Description

**template<typename `_RealType` = double>**  
**class `std::normal_distribution`< `_RealType` >**

A normal continuous distribution for random numbers.  
The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{\sigma^2}}$$

Since

C++11

### 5.703.2 Member Typedef Documentation

#### **result\_type**

template<typename `_RealType` = double>  
typedef `_RealType` `std::normal_distribution`< `_RealType` >::result\_type  
The type of the range of the distribution.

### 5.703.3 Constructor & Destructor Documentation

#### **normal\_distribution()**

template<typename `_RealType` = double>  
`std::normal_distribution`< `_RealType` >::normal\_distribution (  
    result\_type \_\_mean,  
    result\_type \_\_stddev = result\_type(1)) [inline], [explicit]

Constructs a normal distribution with parameters *mean* and standard deviation.

### 5.703.4 Member Function Documentation

#### **max()**

template<typename `_RealType` = double>  
result\_type `std::normal_distribution`< `_RealType` >::max () const [inline]

Returns the least upper bound value of the distribution.

**mean()**

```
template<typename _RealType = double>
_RealType std::normal_distribution<_RealType>::mean () const [inline]
```

Returns the mean of the distribution.

**min()**

```
template<typename _RealType = double>
result_type std::normal_distribution<_RealType>::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

**operator>() [1/2]**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::normal_distribution<_RealType>::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

**operator>() [2/2]**

```
template<typename _RealType>
template<typename _UniformRandomNumberGenerator>
normal_distribution<_RealType>::result_type std::normal_distribution<_RealType>::operator()
(
    _UniformRandomNumberGenerator & __urng,
    const param_type & __param)
```

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

References [std::log\(\)](#), and [std::sqrt\(\)](#).

**param() [1/2]**

```
template<typename _RealType = double>
param_type std::normal_distribution<_RealType>::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _RealType = double>
void std::normal_distribution<_RealType>::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::normal_distribution<_RealType>::reset () [inline]
```

Resets the distribution state.

**stddev()**

```
template<typename _RealType = double>
_RealType std::normal_distribution< _RealType >::stddev () const [inline]
Returns the standard deviation of the distribution.
```

**5.703.5 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::normal_distribution< _RealType1 > & __x) [friend]
Inserts a normal_distribution random number distribution __x into the output stream __os.
```

**Parameters**

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <code>__os</code> | An output stream.                                 |
| <code>__x</code>  | A normal_distribution random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
template<typename _RealType1>
bool operator== (
    const std::normal_distribution< _RealType1 > & __d1,
    const std::normal_distribution< _RealType1 > & __d2) [friend]
```

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::normal_distribution< _RealType1 > & __x) [friend]
Extracts a normal_distribution random number distribution __x from the input stream __is.
```

**Parameters**

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <code>__is</code> | An input stream.                                      |
| <code>__x</code>  | A normal_distribution random number generator engine. |

**Returns**

The input stream with \_\_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.704 std::not\_equal\_to< \_Tp > Struct Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::not\_equal\_to< \_Tp >:

**Public Types**

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

**Public Member Functions**

- constexpr bool **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

**5.704.1 Detailed Description**

```
template<typename _Tp>
struct std::not_equal_to< _Tp >
```

One of the [comparison functors](#).

**5.704.2 Member Typedef Documentation****first\_argument\_type**

```
typedef _Tp std::binary\_function< \_Tp, \_Tp, bool >::first\_argument\_type [inherited]
first_argument_type is the type of the first argument
```

**result\_type**

typedef bool [std::binary\\_function](#)< \_Tp, \_Tp, bool >::result\_type [inherited]  
 result\_type is the return type

**second\_argument\_type**

typedef \_Tp [std::binary\\_function](#)< \_Tp, \_Tp, bool >::second\_argument\_type [inherited]  
 second\_argument\_type is the type of the second argument  
 The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

**5.705 std::ranges::not\_equal\_to Struct Reference**

```
#include <ranges_cmp.h>
```

**Public Types**

- using [is\\_transparent](#)

**Public Member Functions**

- template<typename \_Tp, typename \_Up>  
 requires equality\_comparable\_with<\_Tp, \_Up>  
 constexpr bool **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::declval](#)<\_Tp >()==[std::declval](#)<\_Up >()))

**5.705.1 Detailed Description**

ranges::not\_equal\_to function object type.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

**5.706 std::not\_equal\_to< void > Struct Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::not\_equal\_to< void >:



## Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` [is\\_transparent](#)
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

## Public Member Functions

- template<typename `_Tp`, typename `_Up`>  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept([std::forward](#)<`_Tp`>(`_t`) !=[std::forward](#)<`_Up`>(`_u`))) -> decltype([std::forward](#)<`_Tp`>(`_t`) !=[std::forward](#)<`_Up`>(`_u`))
- constexpr bool **operator()** (const void &`_x`, const void &`_y`) const

### 5.706.1 Detailed Description

One of the [comparison functors](#).

### 5.706.2 Member Typedef Documentation

#### `first_argument_type`

typedef void [std::binary\\_function](#)< void, void, bool >::`first_argument_type`  
`first_argument_type` is the type of the first argument

#### `result_type`

typedef bool [std::binary\\_function](#)< void, void, bool >::`result_type`  
`result_type` is the return type

#### `second_argument_type`

typedef void [std::binary\\_function](#)< void, void, bool >::`second_argument_type`  
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.707 `__gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4>` Struct Template Reference

```
#include <tag_and_trait.hpp>
```



Inheritance diagram for `__gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4 >`:



#### 5.707.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4>
struct __gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4 >
```

A null node updatator, indicating that no node updates are required.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 5.708 \_\_gnu\_pbds::null\_type Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::null\_type:



### 5.708.1 Detailed Description

Represents no type, or absence of type, for template tricks.

In a mapped-policy, indicates that an associative container is a set.

In a list-update policy, indicates that each link does not need metadata.

In a hash policy, indicates that the combining hash function is actually a ranged hash function.

In a probe policy, indicates that the combining probe function is actually a ranged probe function.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.709 std::experimental::fundamentals\_v1::nullopt\_t Struct Reference

```
#include <optional>
```

### Public Types

- enum class **\_Construct** { **\_Token** }

## Public Member Functions

- constexpr **nullopt\_t** (\_Construct)

### 5.709.1 Detailed Description

Tag type to disengage optional objects.

The documentation for this struct was generated from the following file:

- [experimental/optional](#)

## 5.710 std::num\_get<\_CharT, \_InIter> Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::num\_get<\_CharT, \_InIter>:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_InIter` **iter\_type**

## Public Member Functions

- **num\_get** (size\_t \_\_refs=0)
- template<typename \_ValueT>  
\_InIter **M\_extract\_int** (\_InIter \_\_beg, \_InIter \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, \_ValueT &\_\_v) const
- **iter\_type get** (iter\_type \_\_in, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, bool &\_\_v) const
- **iter\_type get** (iter\_type \_\_in, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, void \*&\_\_v) const
- **iter\_type get** (iter\_type \_\_in, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, long &\_\_v) const
- **iter\_type get** (iter\_type \_\_in, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, unsigned short &\_\_v) const
- **iter\_type get** (iter\_type \_\_in, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, unsigned int &\_\_v) const
- **iter\_type get** (iter\_type \_\_in, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, unsigned long &\_\_v) const
- **iter\_type get** (iter\_type \_\_in, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, long long &\_\_v) const

- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, double & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long double & __v) const`

### Static Public Attributes

- static `locale::id id`

### Protected Member Functions

- virtual `~num_get ()`
- `iter_type _M_extract_float (iter_type, iter_type, ios_base &, ios_base::iostate &, string &) const`
- `template<typename _ValueT>  
iter_type _M_extract_int (iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &) const`
- `template<typename _CharT2>  
__gnu_cxx::__enable_if<__is_char<_CharT2>>::__value, int >::__type _M_find (const _CharT2 *, size_t __len, _CharT2 __c) const`
- `template<typename _CharT2>  
__gnu_cxx::__enable_if<!__is_char<_CharT2>>::__value, int >::__type _M_find (const _CharT2 * __zero, size_t __len, _CharT2 __c) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, long double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, void *&) const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale & __cloc) throw ()`
- static void `_S_create_c_locale (__c_locale & __cloc, const char * __s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale & __cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char * __s)`

### 5.710.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::num_get< _CharT, _InIter >
```

Primary class template `num_get`.

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

The `num_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `num_get` facet.

### 5.710.2 Member Typedef Documentation

#### `char_type`

```
template<typename _CharT, typename _InIter>
typedef _CharT std::num_get< _CharT, _InIter >::char_type
Public typedefs.
```

#### `iter_type`

```
template<typename _CharT, typename _InIter>
typedef _InIter std::num_get< _CharT, _InIter >::iter_type
Public typedefs.
```

### 5.710.3 Constructor & Destructor Documentation

#### `num_get()`

```
template<typename _CharT, typename _InIter>
std::num_get< _CharT, _InIter >::num_get (
    size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

#### `~num_get()`

```
template<typename _CharT, typename _InIter>
virtual std::num_get< _CharT, _InIter >::~num_get () [inline], [protected], [virtual]
Destructor.
```

### 5.710.4 Member Function Documentation

#### `do_get()` [1/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
```

```
ios_base::iostate & __err,
long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

#### do\_get() [2/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get<_CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

**do\_get()** [3/11]

```
template<typename _CharT, typename _InIter>
virtual iter\_type std::num\_get< _CharT, _InIter >::do_get (
    iter\_type __beg,
    iter\_type __end,
    ios\_base & __io,
    ios\_base::iostate & __err,
    unsigned int & __v) const \[inline\], \[protected\], \[virtual\]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                       |                             |
|-----------------------|-----------------------------|
| <a href="#">__beg</a> | Start of input stream.      |
| <a href="#">__end</a> | End of input stream.        |
| <a href="#">__io</a>  | Source of locale and flags. |
| <a href="#">__err</a> | Error flags to set.         |
| <a href="#">__v</a>   | Value to format and insert. |

**Returns**

Iterator after reading.

**do\_get()** [4/11]

```
template<typename _CharT, typename _InIter>
virtual iter\_type std::num\_get< _CharT, _InIter >::do_get (
    iter\_type __beg,
    iter\_type __end,
    ios\_base & __io,
    ios\_base::iostate & __err,
    unsigned long & __v) const \[inline\], \[protected\], \[virtual\]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                       |                             |
|-----------------------|-----------------------------|
| <a href="#">__beg</a> | Start of input stream.      |
| <a href="#">__end</a> | End of input stream.        |
| <a href="#">__io</a>  | Source of locale and flags. |
| <a href="#">__err</a> | Error flags to set.         |
| <a href="#">__v</a>   | Value to format and insert. |

**Returns**

Iterator after reading.

**do\_get()** [5/11]

```
template<typename _CharT, typename _InIter>
virtual iter\_type std::num_get< _CharT, _InIter >::do_get (
    iter\_type __beg,
    iter\_type __end,
    ios\_base & __io,
    ios\_base::iostate & __err,
    unsigned long long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                       |                             |
|-----------------------|-----------------------------|
| <a href="#">__beg</a> | Start of input stream.      |
| <a href="#">__end</a> | End of input stream.        |
| <a href="#">__io</a>  | Source of locale and flags. |
| <a href="#">__err</a> | Error flags to set.         |
| <a href="#">__v</a>   | Value to format and insert. |

**Returns**

Iterator after reading.

**do\_get()** [6/11]

```
template<typename _CharT, typename _InIter>
virtual iter\_type std::num_get< _CharT, _InIter >::do_get (
    iter\_type __beg,
    iter\_type __end,
    ios\_base & __io,
    ios\_base::iostate & __err,
    unsigned short & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                       |                             |
|-----------------------|-----------------------------|
| <a href="#">__beg</a> | Start of input stream.      |
| <a href="#">__end</a> | End of input stream.        |
| <a href="#">__io</a>  | Source of locale and flags. |



|                    |                             |
|--------------------|-----------------------------|
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

#### **do\_get()** [7/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    bool & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

#### See also

`get()` for more details.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

References `std::ios_base::_M_getloc()`, `std::ios_base::boolalpha`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::flags()`, and `std::ios_base::goodbit`.

Referenced by `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`, `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`, `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`, `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`, `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`, `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`, `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`, `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`, `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`, and `std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get()`.

#### **do\_get()** [8/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
```

```

        ios_base::iostate & __err,
        double & __v) const [protected], [virtual]

```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::ios\\_base::eofbit](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::r](#)

#### `do_get()` [9/11]

```

template<typename _CharT, typename _InIter>
_InIter std::num_get<_CharT, _InIter>::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    float & __v) const [protected], [virtual]

```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::ios\\_base::eofbit](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::r](#)

**do\_get()** [10/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long double & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::c\\_str\(\)](#), [std::ios\\_base::eofbit](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::r](#)

**do\_get()** [11/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    void *& __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

References [std::ios\\_base::basefield](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::fmtflags](#), and [std::ios\\_base::hex](#).

**get()** [1/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    bool & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling `num_get::do_get()`.

If `ios_base::boolalpha` is set, attempts to read `ctype<CharT>::truenamename()` or `ctype<CharT>::falsename()`. Sets *v* to true or false if successful. Sets *err* to `ios_base::failbit` if reading the string fails. Sets *err* to `ios_base::eofbit` if the stream is emptied.

If `ios_base::boolalpha` is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set *err* to `ios_base::failbit`.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#).

**get()** [2/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    double & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [3/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    float & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`. If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [4/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

#### get() [5/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long double & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.

The decimal point character used is numpunct::decimal\_point(). Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

**get()** [6/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [7/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned int & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

**get()** [8/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

**get()** [9/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
```



```
ios_base::iostate & __err,
unsigned long long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

#### `get()` [10/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned short & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [11/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    void *& __v) const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf p` specifier.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**5.710.5 Member Data Documentation****id**

```
template<typename _CharT, typename _InIter>
locale::id std::num_get< _CharT, _InIter >::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

**5.711 std::num\_put<\_CharT, \_OutIter > Class Template Reference**

```
#include <locale_facets.h>
```

Inheritance diagram for `std::num_put<_CharT, _OutIter >`:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_OutIter` `iter_type`

## Public Member Functions

- `num_put` (`size_t` \_\_refs=0)
- template<typename `_ValueT`>  
`_OutIter _M_insert_float` (`_OutIter` \_\_s, `ios_base` &\_\_io, `_CharT` \_\_fill, `char` \_\_mod, `_ValueT` \_\_v) const
- template<typename `_ValueT`>  
`_OutIter _M_insert_int` (`_OutIter` \_\_s, `ios_base` &\_\_io, `_CharT` \_\_fill, `_ValueT` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `bool` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `const void *`\_\_v) const
  
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `long` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `unsigned long` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `long long` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `unsigned long long` \_\_v) const
  
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `double` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `long double` \_\_v) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~num_put` ()
- void `_M_group_float` (`const char *`\_\_grouping, `size_t` \_\_grouping\_size, `char_type` \_\_sep, `const char_type *`\_\_p, `char_type *`\_\_new, `char_type *`\_\_cs, `int` &\_\_len) const

- void **\_M\_group\_int** (const char \*\_\_grouping, size\_t \_\_grouping\_size, char\_type \_\_sep, ios\_base &\_\_io, char\_type \*\_\_new, char\_type \*\_\_cs, int &\_\_len) const
- template<typename \_ValueT>  
iter\_type **\_M\_insert\_float** (iter\_type, ios\_base &\_\_io, char\_type \_\_fill, char \_\_mod, \_ValueT \_\_v) const
- template<typename \_ValueT>  
iter\_type **\_M\_insert\_int** (iter\_type, ios\_base &\_\_io, char\_type \_\_fill, \_ValueT \_\_v) const
- void **\_M\_pad** (char\_type \_\_fill, streamsize \_\_w, ios\_base &\_\_io, char\_type \*\_\_new, const char\_type \*\_\_cs, int &\_\_len) const
- virtual iter\_type **do\_put** (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, bool \_\_v) const
- virtual iter\_type **do\_put** (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, long \_\_v) const
- virtual iter\_type **do\_put** (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, unsigned long \_\_v) const
- virtual iter\_type **do\_put** (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, long long \_\_v) const
- virtual iter\_type **do\_put** (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, unsigned long long \_\_v) const
- virtual iter\_type **do\_put** (iter\_type, ios\_base &, char\_type, double) const
- virtual iter\_type **do\_put** (iter\_type, ios\_base &, char\_type, long double) const
- virtual iter\_type **do\_put** (iter\_type, ios\_base &, char\_type, const void \*) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_type\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 5.711.1 Detailed Description

```
template<typename _CharT, typename _OutIter>
class std::num_put<_CharT, _OutIter>
```

Primary class template num\_put.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators. The num\_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num\_put facet.

#### 5.711.2 Member Typedef Documentation

##### char\_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::num_put<_CharT, _OutIter>::char_type
```

Public typedefs.

##### iter\_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::num_put<_CharT, _OutIter>::iter_type
```

Public typedefs.

### 5.711.3 Constructor & Destructor Documentation

#### num\_put()

```
template<typename _CharT, typename _OutIter>
std::num_put< _CharT, _OutIter >::num_put (
    size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

#### ~num\_put()

```
template<typename _CharT, typename _OutIter>
virtual std::num_put< _CharT, _OutIter >::~~num_put () [inline], [protected], [virtual]
```

Destructor.

### 5.711.4 Member Function Documentation

#### do\_put() [1/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    bool __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

#### Returns

Iterator after writing.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::ios\\_base::adjustfield](#), [std::ios\\_base::boolalpha](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::fmtflags](#), [std::ios\\_base::left](#), and [std::ios\\_base::width\(\)](#).

Referenced by [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), and [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#).

**do\_put()** [2/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**do\_put()** [3/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**do\_put()** [4/8]

```
template<typename _CharT, typename _OutIter>
virtual iter\_type std::num_put< _CharT, _OutIter >::do_put (
    iter\_type __s,
    ios\_base & __io,
    char\_type __fill,
    unsigned long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                             |                               |
|-----------------------------|-------------------------------|
| <a href="#">↵</a><br>__s    | Stream to write to.           |
| <a href="#">↵</a><br>__io   | Source of locale and flags.   |
| <a href="#">↵</a><br>__fill | Char_type to use for filling. |
| <a href="#">↵</a><br>__v    | Value to format and insert.   |

**Returns**

Iterator after writing.

**do\_put()** [5/8]

```
template<typename _CharT, typename _OutIter>
virtual iter\_type std::num_put< _CharT, _OutIter >::do_put (
    iter\_type __s,
    ios\_base & __io,
    char\_type __fill,
    unsigned long long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                             |                               |
|-----------------------------|-------------------------------|
| <a href="#">↵</a><br>__s    | Stream to write to.           |
| <a href="#">↵</a><br>__io   | Source of locale and flags.   |
| <a href="#">↵</a><br>__fill | Char_type to use for filling. |
| <a href="#">↵</a><br>__v    | Value to format and insert.   |

**Returns**

Iterator after writing.

**do\_put()** [6/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const void * __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

References [std::ios\\_base::basefield](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::fmtflags](#), [std::ios\\_base::hex](#), [std::ios\\_base::showbase](#), and [std::ios\\_base::uppercase](#).

**do\_put()** [7/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    double __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |



**Returns**

Iterator after writing.

**do\_put()** [8/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long double __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**put()** [1/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    bool __v) const [inline]
```

Numeric formatting.

Formats the boolean `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

If `ios_base::boolalpha` is set, writes `ctype<CharT>::truename()` or `ctype<CharT>::falsename()`. Otherwise formats `v` as an int.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits >::operator<<\(\)](#).

**put()** [2/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put<_CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const void * __v) const [inline]
```

Numeric formatting.

Formats the pointer value *v* and inserts it into a stream. It does so by calling num\_put::do\_put().

This function formats *v* as an unsigned long with ios\_base::hex and ios\_base::showbase set.

**Parameters**

|                               |                               |
|-------------------------------|-------------------------------|
| <a href="#"><i>__s</i></a>    | Stream to write to.           |
| <a href="#"><i>__io</i></a>   | Source of locale and flags.   |
| <a href="#"><i>__fill</i></a> | Char_type to use for filling. |
| <a href="#"><i>__v</i></a>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**put()** [3/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put<_CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    double __v) const [inline]
```

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling num\_put::do\_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios\_base::floatfield. If equal to ios\_base::fixed, formats like the printf f specifier. Else if equal to ios\_base::scientific, formats like e or E with ios\_base::uppercase unset or set respectively. Otherwise, formats like g or G depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like g or G.

The output precision is given by io.precision(). This precision is capped at numeric\_limits::digits10 + 2 (different for double and long double). The default precision is 6.

If ios\_base::showpos is set, '+' is output before positive values. If ios\_base::showpoint is set, a decimal point will always be output.

The decimal point character used is `numunct::decimal_point()`. Thousands separators are inserted according to `numunct::grouping()` and `numunct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_↵base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

#### Parameters

|                          |                               |
|--------------------------|-------------------------------|
| <code>↵<br/>_s</code>    | Stream to write to.           |
| <code>↵<br/>_io</code>   | Source of locale and flags.   |
| <code>↵<br/>_fill</code> | Char_type to use for filling. |
| <code>↵<br/>_v</code>    | Value to format and insert.   |

#### Returns

Iterator after writing.

#### put() [4/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long __v) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `numunct::decimal_point()`. Thousands separators are inserted according to `numunct::grouping()` and `numunct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_↵base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

#### Parameters

|                          |                               |
|--------------------------|-------------------------------|
| <code>↵<br/>_s</code>    | Stream to write to.           |
| <code>↵<br/>_io</code>   | Source of locale and flags.   |
| <code>↵<br/>_fill</code> | Char_type to use for filling. |

|                  |                             |
|------------------|-----------------------------|
| <code>__v</code> | Value to format and insert. |
|------------------|-----------------------------|

**Returns**

Iterator after writing.

**put()** [5/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long double __v) const [inline]
```

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling num\_put::do\_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios\_base::floatfield. If equal to ios\_base::fixed, formats like the printf f specifier. Else if equal to ios\_base::scientific, formats like e or E with ios\_base::uppercase unset or set respectively. Otherwise, formats like g or G depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like g or G.

The output precision is given by io.precision(). This precision is capped at numeric\_limits::digits10 + 2 (different for double and long double). The default precision is 6.

If ios\_base::showpos is set, '+' is output before positive values. If ios\_base::showpoint is set, a decimal point will always be output.

The decimal point character used is numpunct::decimal\_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands\_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios\_base::adjustfield) == ios\_base::left, result is padded at the end. If ios\_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**put()** [6/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
```

```

iter_type __s,
ios_base & __io,
char_type __fill,
long long __v) const [inline]

```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

#### Parameters

|                           |                               |
|---------------------------|-------------------------------|
| <code>↵<br/>__s</code>    | Stream to write to.           |
| <code>↵<br/>__io</code>   | Source of locale and flags.   |
| <code>↵<br/>__fill</code> | Char_type to use for filling. |
| <code>↵<br/>__v</code>    | Value to format and insert.   |

#### Returns

Iterator after writing.

#### put() [7/8]

```

template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    unsigned long __v) const [inline]

```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

#### Returns

Iterator after writing.

#### put() [8/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    unsigned long long __v) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**5.711.5 Member Data Documentation****id**

```
template<typename _CharT, typename _OutIter>
locale::id std::num_put< _CharT, _OutIter >::id [static]
Numpunct facet id.
```

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

**5.712 std::numeric\_limits< \_Tp > Struct Template Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< \_Tp >:

**Static Public Member Functions**

- static constexpr \_Tp [denorm\\_min](#) () noexcept
- static constexpr \_Tp [epsilon](#) () noexcept
- static constexpr \_Tp [infinity](#) () noexcept
- static constexpr \_Tp [lowest](#) () noexcept
- static constexpr \_Tp [max](#) () noexcept
- static constexpr \_Tp [min](#) () noexcept
- static constexpr \_Tp [quiet\\_NaN](#) () noexcept
- static constexpr \_Tp [round\\_error](#) () noexcept
- static constexpr \_Tp [signaling\\_NaN](#) () noexcept

**Static Public Attributes**

- static constexpr int [digits](#)
- static constexpr int [digits10](#)
- static constexpr float\_denorm\_style [has\\_denorm](#)
- static constexpr bool [has\\_denorm\\_loss](#)

- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)
- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)
- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

### 5.712.1 Detailed Description

**template<typename \_Tp>**  
**struct `std::numeric_limits<_Tp>`**

Properties of fundamental types.

This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

### 5.712.2 Member Function Documentation

#### **denorm\_min()**

```
template<typename _Tp>
static constexpr _Tp std::numeric\_limits<\_Tp>::denorm\_min () [inline], [static], [constexpr],
[noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### **epsilon()**

```
template<typename _Tp>
static constexpr _Tp std::numeric\_limits<\_Tp>::epsilon () [inline], [static], [constexpr],
[noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

Referenced by [std::generate\\_canonical\(\)](#), [std::binomial\\_distribution<\\_IntType>::operator\(\)\(\)](#), [std::poisson\\_distribution<\\_IntType>::operator\(\)\(\)](#) and [std::operator<<\(\)](#).

#### **infinity()**

```
template<typename _Tp>
static constexpr _Tp std::numeric\_limits<\_Tp>::infinity () [inline], [static], [constexpr],
[noexcept]
```

The representation of positive infinity, if `has_infinity`.



**lowest()**

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::lowest () [inline], [static], [constexpr], [noexcept]
```

A finite value  $x$  such that there is no other finite value  $y$  where  $y < x$ .

Referenced by [std::cauchy\\_distribution< \\_RealType >::min\(\)](#), [std::extreme\\_value\\_distribution< \\_RealType >::min\(\)](#), [std::normal\\_distribution< \\_RealType >::min\(\)](#), and [std::student\\_t\\_distribution< \\_RealType >::min\(\)](#).

**max()**

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::max () [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [std::bernoulli\\_distribution::max\(\)](#), [std::cauchy\\_distribution< \\_RealType >::max\(\)](#), [std::chi\\_squared\\_distribution< \\_RealType >::max\(\)](#), [std::exponential\\_distribution< \\_RealType >::max\(\)](#), [std::extreme\\_value\\_distribution< \\_RealType >::max\(\)](#), [std::fisher\\_f\\_distribution< \\_RealType >::max\(\)](#), [std::gamma\\_distribution< \\_RealType >::max\(\)](#), [std::geometric\\_distribution< \\_IntType >::max\(\)](#), [std::lognormal\\_distribution< \\_RealType >::max\(\)](#), [std::negative\\_binomial\\_distribution< \\_IntType >::max\(\)](#), [std::normal\\_distribution< \\_RealType >::max\(\)](#), [std::poisson\\_distribution< \\_IntType >::max\(\)](#), [std::student\\_t\\_distribution< \\_RealType >::max\(\)](#), [std::weibull\\_distribution< \\_RealType >::max\(\)](#), [std::binomial\\_distribution< \\_IntType >::operator\(\)](#), [std::independent\\_bits\\_engine< \\_RandomNumberEngine, \\_\\_w, \\_UIntType >::operator\(\)](#), [std::poisson\\_distribution< \\_IntType >::operator\(\)](#), and [std::operator<<\(\)](#).

**min()**

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::min () [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Referenced by [std::bernoulli\\_distribution::min\(\)](#).

**quiet\_NaN()**

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::quiet_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

**round\_error()**

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::round_error () [inline], [static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

**5.712.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr], [inherited]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Referenced by `std::generate_canonical()`, `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::operator()`, `std::operator<<()`, and `std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::seed()`.

### **digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr], [inherited]
```

The number of base 10 digits that can be represented without change.

### **has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr], [inherited]
```

See `std::float_denorm_style` for more information.

### **has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr], [inherited]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

### **has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr], [inherited]
```

True if the type has a representation for positive infinity.

### **has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr], [inherited]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

### **has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr], [inherited]
```

True if the type has a representation for a signaling Not a Number.

### **is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr], [inherited]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

### **is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr], [inherited]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

### **is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr], [inherited]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr], [inherited]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr], [inherited]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr], [inherited]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr], [inherited]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr], [inherited]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr], [inherited]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr], [inherited]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr], [inherited]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr], [inherited]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr], [inherited]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr], [inherited]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr], [inherited]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr], [inherited]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.713 std::numeric\_limits< bool > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< bool >:

**Static Public Member Functions**

- static constexpr bool [denorm\\_min](#) () noexcept
- static constexpr bool **denorm\_min** () noexcept
- static constexpr bool [epsilon](#) () noexcept
- static constexpr bool **epsilon** () noexcept
- static constexpr bool [infinity](#) () noexcept
- static constexpr bool **infinity** () noexcept
- static constexpr bool [lowest](#) () noexcept
- static constexpr bool **lowest** () noexcept
- static constexpr bool [max](#) () noexcept
- static constexpr bool **max** () noexcept
- static constexpr bool [min](#) () noexcept
- static constexpr bool **min** () noexcept

- static constexpr bool [quiet\\_NaN](#) () noexcept
- static constexpr bool **quiet\_NaN** () noexcept
- static constexpr bool [round\\_error](#) () noexcept
- static constexpr bool **round\_error** () noexcept
- static constexpr bool [signaling\\_NaN](#) () noexcept
- static constexpr bool **signaling\_NaN** () noexcept

#### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**

- static constexpr bool `tinyness_before`
- static constexpr bool `tinyness_before`
- static constexpr bool `traps`
- static constexpr bool `traps`

### 5.713.1 Detailed Description

`numeric_limits<bool>` specialization.

### 5.713.2 Member Function Documentation

#### `denorm_min()`

```
static constexpr bool std::numeric_limits< bool >::denorm_min () [inline], [static], [constexpr],  
[noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### `epsilon()`

```
static constexpr bool std::numeric_limits< bool >::epsilon () [inline], [static], [constexpr],  
[noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### `infinity()`

```
static constexpr bool std::numeric_limits< bool >::infinity () [inline], [static], [constexpr],  
[noexcept]
```

The representation of positive infinity, if `has_infinity`.

#### `lowest()`

```
static constexpr bool std::numeric_limits< bool >::lowest () [inline], [static], [constexpr],  
[noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

#### `max()`

```
static constexpr bool std::numeric_limits< bool >::max () [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

#### `min()`

```
static constexpr bool std::numeric_limits< bool >::min () [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

#### `quiet_NaN()`

```
static constexpr bool std::numeric_limits< bool >::quiet_NaN () [inline], [static], [constexpr],  
[noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

**round\_error()**

```
static constexpr bool std::numeric_limits< bool >::round_error () [inline], [static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr bool std::numeric_limits< bool >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

**5.713.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving `+`, `-`, or `*` on values of that type whose result would fall outside the range `[min(),max())`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.



**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.714 std::numeric\_limits< char > Struct Reference**

```
#include <limits>
```

Inheritance diagram for `std::numeric_limits< char >`:



### Static Public Member Functions

- static constexpr char [denorm\\_min](#) () noexcept
- static constexpr char **denorm\_min** () noexcept
- static constexpr char [epsilon](#) () noexcept
- static constexpr char **epsilon** () noexcept
- static constexpr char [infinity](#) () noexcept
- static constexpr char **infinity** () noexcept
- static constexpr char [lowest](#) () noexcept
- static constexpr char **lowest** () noexcept
- static constexpr char [max](#) () noexcept
- static constexpr char **max** () noexcept
- static constexpr char [min](#) () noexcept
- static constexpr char **min** () noexcept
- static constexpr char [quiet\\_NaN](#) () noexcept
- static constexpr char **quiet\_NaN** () noexcept
- static constexpr char [round\\_error](#) () noexcept
- static constexpr char **round\_error** () noexcept
- static constexpr char [signaling\\_NaN](#) () noexcept
- static constexpr char **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)

- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

#### 5.714.1 Detailed Description

`numeric_limits<char>` specialization.

#### 5.714.2 Member Function Documentation

##### **denorm\_min()**

```
static constexpr char std::numeric\_limits< char >::denorm\_min \(\) [inline], [static], [constexpr],  
[noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

##### **epsilon()**

```
static constexpr char std::numeric\_limits< char >::epsilon \(\) [inline], [static], [constexpr],  
[noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

##### **infinity()**

```
static constexpr char std::numeric\_limits< char >::infinity \(\) [inline], [static], [constexpr],  
[noexcept]
```

The representation of positive infinity, if `has_infinity`.

##### **lowest()**

```
static constexpr char std::numeric\_limits< char >::lowest \(\) [inline], [static], [constexpr],  
[noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

##### **max()**

```
static constexpr char std::numeric\_limits< char >::max \(\) [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

**min()**

static constexpr char `std::numeric_limits< char >::min ()` [inline], [static], [constexpr], [noexcept]  
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

static constexpr char `std::numeric_limits< char >::quiet_NaN ()` [inline], [static], [constexpr], [noexcept]  
The representation of a quiet Not a Number, if `has_quiet_NaN`.

**round\_error()**

static constexpr char `std::numeric_limits< char >::round_error ()` [inline], [static], [constexpr], [noexcept]  
The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

static constexpr char `std::numeric_limits< char >::signaling_NaN ()` [inline], [static], [constexpr], [noexcept]  
The representation of a signaling Not a Number, if `has_signaling_NaN`.

**5.714.3 Member Data Documentation****digits**

int `std::__numeric_limits_base::digits` [static], [constexpr]  
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

int `std::__numeric_limits_base::digits10` [static], [constexpr]  
The number of base 10 digits that can be represented without change.

**has\_denorm**

`float_denorm_style` `std::__numeric_limits_base::has_denorm` [static], [constexpr]  
See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

bool `std::__numeric_limits_base::has_denorm_loss` [static], [constexpr]  
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

bool `std::__numeric_limits_base::has_infinity` [static], [constexpr]  
True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

bool `std::__numeric_limits_base::has_quiet_NaN` [static], [constexpr]  
True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.715 std::numeric\_limits< char16\_t > Struct Reference**

```
#include <limits>
```

Inheritance diagram for `std::numeric_limits< char16_t >`:



### Static Public Member Functions

- static constexpr char16\_t [denorm\\_min](#) () noexcept
- static constexpr char16\_t **denorm\_min** () noexcept
- static constexpr char16\_t [epsilon](#) () noexcept
- static constexpr char16\_t **epsilon** () noexcept
- static constexpr char16\_t [infinity](#) () noexcept
- static constexpr char16\_t **infinity** () noexcept
- static constexpr char16\_t [lowest](#) () noexcept
- static constexpr char16\_t **lowest** () noexcept
- static constexpr char16\_t [max](#) () noexcept
- static constexpr char16\_t **max** () noexcept
- static constexpr char16\_t [min](#) () noexcept
- static constexpr char16\_t **min** () noexcept
- static constexpr char16\_t [quiet\\_NaN](#) () noexcept
- static constexpr char16\_t **quiet\_NaN** () noexcept
- static constexpr char16\_t [round\\_error](#) () noexcept
- static constexpr char16\_t **round\_error** () noexcept
- static constexpr char16\_t [signaling\\_NaN](#) () noexcept
- static constexpr char16\_t **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)

- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.715.1 Detailed Description

`numeric_limits<char16_t>` specialization.

### 5.715.2 Member Function Documentation

#### **denorm\_min()**

```
static constexpr char16_t std::numeric_limits< char16_t >::denorm_min () [inline], [static],
[constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### **epsilon()**

```
static constexpr char16_t std::numeric_limits< char16_t >::epsilon () [inline], [static], [constexpr],
[noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.



**infinity()**

```
static constexpr char16_t std::numeric_limits< char16_t >::infinity () [inline], [static], [constexpr],  
[noexcept]
```

The representation of positive infinity, if has\_infinity.

**lowest()**

```
static constexpr char16_t std::numeric_limits< char16_t >::lowest () [inline], [static], [constexpr],  
[noexcept]
```

A finite value x such that there is no other finite value y where  $y < x$ .

**max()**

```
static constexpr char16_t std::numeric_limits< char16_t >::max () [inline], [static], [constexpr],  
[noexcept]
```

The maximum finite value.

**min()**

```
static constexpr char16_t std::numeric_limits< char16_t >::min () [inline], [static], [constexpr],  
[noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

```
static constexpr char16_t std::numeric_limits< char16_t >::quiet_NaN () [inline], [static],  
[constexpr], [noexcept]
```

The representation of a quiet Not a Number, if has\_quiet\_NaN.

**round\_error()**

```
static constexpr char16_t std::numeric_limits< char16_t >::round_error () [inline], [static],  
[constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr char16_t std::numeric_limits< char16_t >::signaling_NaN () [inline], [static],  
[constexpr], [noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.

**5.715.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving `+`, `-`, or `*` on values of that type whose result would fall outside the range `[min(),max())`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.716 std::numeric\_limits< char32\_t > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< char32\_t >:

**Static Public Member Functions**

- static constexpr char32\_t [denorm\\_min](#) () noexcept
- static constexpr char32\_t **denorm\_min** () noexcept
- static constexpr char32\_t [epsilon](#) () noexcept
- static constexpr char32\_t **epsilon** () noexcept
- static constexpr char32\_t [infinity](#) () noexcept
- static constexpr char32\_t **infinity** () noexcept
- static constexpr char32\_t [lowest](#) () noexcept
- static constexpr char32\_t **lowest** () noexcept
- static constexpr char32\_t [max](#) () noexcept
- static constexpr char32\_t **max** () noexcept
- static constexpr char32\_t [min](#) () noexcept
- static constexpr char32\_t **min** () noexcept
- static constexpr char32\_t [quiet\\_NaN](#) () noexcept
- static constexpr char32\_t **quiet\_NaN** () noexcept
- static constexpr char32\_t [round\\_error](#) () noexcept
- static constexpr char32\_t **round\_error** () noexcept
- static constexpr char32\_t [signaling\\_NaN](#) () noexcept
- static constexpr char32\_t **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.716.1 Detailed Description

numeric\_limits<char32\_t> specialization.

### 5.716.2 Member Function Documentation

#### denorm\_min()

```
static constexpr char32_t std::numeric_limits< char32_t >::denorm_min () [inline], [static],  
[constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### epsilon()

```
static constexpr char32_t std::numeric_limits< char32_t >::epsilon () [inline], [static], [constexpr],  
[noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### infinity()

```
static constexpr char32_t std::numeric_limits< char32_t >::infinity () [inline], [static], [constexpr],  
[noexcept]
```

The representation of positive infinity, if `has_infinity`.

#### lowest()

```
static constexpr char32_t std::numeric_limits< char32_t >::lowest () [inline], [static], [constexpr],  
[noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

#### max()

```
static constexpr char32_t std::numeric_limits< char32_t >::max () [inline], [static], [constexpr],  
[noexcept]
```

The maximum finite value.

#### min()

```
static constexpr char32_t std::numeric_limits< char32_t >::min () [inline], [static], [constexpr],  
[noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

#### quiet\_NaN()

```
static constexpr char32_t std::numeric_limits< char32_t >::quiet_NaN () [inline], [static],  
[constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

#### round\_error()

```
static constexpr char32_t std::numeric_limits< char32_t >::round_error () [inline], [static],  
[constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

#### signaling\_NaN()

```
static constexpr char32_t std::numeric_limits< char32_t >::signaling_NaN () [inline], [static],  
[constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

### 5.716.3 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that *radix* raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that *radix* raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.



**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.717 std::numeric\_limits< double > Struct Reference**

```
#include <limits>
```

Inheritance diagram for `std::numeric_limits< double >`:

**Static Public Member Functions**

- static constexpr double [denorm\\_min](#) () noexcept
- static constexpr double **denorm\_min** () noexcept
- static constexpr double [epsilon](#) () noexcept
- static constexpr double **epsilon** () noexcept
- static constexpr double [infinity](#) () noexcept
- static constexpr double **infinity** () noexcept
- static constexpr double [lowest](#) () noexcept
- static constexpr double **lowest** () noexcept
- static constexpr double [max](#) () noexcept
- static constexpr double **max** () noexcept
- static constexpr double [min](#) () noexcept
- static constexpr double **min** () noexcept

- static constexpr double [quiet\\_NaN](#) () noexcept
- static constexpr double **quiet\_NaN** () noexcept
- static constexpr double [round\\_error](#) () noexcept
- static constexpr double **round\_error** () noexcept
- static constexpr double [signaling\\_NaN](#) () noexcept
- static constexpr double **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**

- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.717.1 Detailed Description

numeric\_limits<double> specialization.

### 5.717.2 Member Function Documentation

#### denorm\_min()

```
static constexpr double std::numeric_limits< double >::denorm_min () [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### epsilon()

```
static constexpr double std::numeric_limits< double >::epsilon () [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### infinity()

```
static constexpr double std::numeric_limits< double >::infinity () [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

#### lowest()

```
static constexpr double std::numeric_limits< double >::lowest () [inline], [static], [constexpr], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

#### max()

```
static constexpr double std::numeric_limits< double >::max () [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

#### min()

```
static constexpr double std::numeric_limits< double >::min () [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

#### quiet\_NaN()

```
static constexpr double std::numeric_limits< double >::quiet_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

**round\_error()**

```
static constexpr double std::numeric_limits< double >::round_error () [inline], [static], [constexpr],  
[noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr double std::numeric_limits< double >::signaling_NaN () [inline], [static],  
[constexpr], [noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.

**5.717.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that radix raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.718 std::numeric\_limits< float > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< float >:



### Static Public Member Functions

- static constexpr float [denorm\\_min](#) () noexcept
- static constexpr float **denorm\_min** () noexcept
- static constexpr float [epsilon](#) () noexcept
- static constexpr float **epsilon** () noexcept
- static constexpr float [infinity](#) () noexcept
- static constexpr float **infinity** () noexcept
- static constexpr float [lowest](#) () noexcept
- static constexpr float **lowest** () noexcept
- static constexpr float [max](#) () noexcept
- static constexpr float **max** () noexcept
- static constexpr float [min](#) () noexcept
- static constexpr float **min** () noexcept
- static constexpr float [quiet\\_NaN](#) () noexcept
- static constexpr float **quiet\_NaN** () noexcept
- static constexpr float [round\\_error](#) () noexcept
- static constexpr float **round\_error** () noexcept
- static constexpr float [signaling\\_NaN](#) () noexcept
- static constexpr float **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)

- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.718.1 Detailed Description

`numeric_limits<float>` specialization.

### 5.718.2 Member Function Documentation

#### **denorm\_min()**

```
static constexpr float std::numeric\_limits< float >::denorm\_min () [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### **epsilon()**

```
static constexpr float std::numeric\_limits< float >::epsilon () [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### **infinity()**

```
static constexpr float std::numeric\_limits< float >::infinity () [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

#### **lowest()**

```
static constexpr float std::numeric\_limits< float >::lowest () [inline], [static], [constexpr], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

#### **max()**

```
static constexpr float std::numeric\_limits< float >::max () [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.



**min()**

```
static constexpr float std::numeric_limits< float >::min () [inline], [static], [constexpr],  
[noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

```
static constexpr float std::numeric_limits< float >::quiet_NaN () [inline], [static], [constexpr],  
[noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

**round\_error()**

```
static constexpr float std::numeric_limits< float >::round_error () [inline], [static], [constexpr],  
[noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr float std::numeric_limits< float >::signaling_NaN () [inline], [static], [constexpr],  
[noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

**5.718.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving `+`, `-`, or `*` on values of that type whose result would fall outside the range `[min(),max())`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.719 std::numeric\_limits< int > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< int >:



### Static Public Member Functions

- static constexpr int [denorm\\_min](#) () noexcept
- static constexpr int **denorm\_min** () noexcept
- static constexpr int [epsilon](#) () noexcept
- static constexpr int **epsilon** () noexcept
- static constexpr int [infinity](#) () noexcept
- static constexpr int **infinity** () noexcept
- static constexpr int [lowest](#) () noexcept
- static constexpr int **lowest** () noexcept
- static constexpr int [max](#) () noexcept
- static constexpr int **max** () noexcept
- static constexpr int [min](#) () noexcept
- static constexpr int **min** () noexcept
- static constexpr int [quiet\\_NaN](#) () noexcept
- static constexpr int **quiet\_NaN** () noexcept
- static constexpr int [round\\_error](#) () noexcept
- static constexpr int **round\_error** () noexcept
- static constexpr int [signaling\\_NaN](#) () noexcept
- static constexpr int **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)

- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.719.1 Detailed Description

`numeric_limits<int>` specialization.

### 5.719.2 Member Function Documentation

#### **denorm\_min()**

```
static constexpr int std::numeric\_limits< int >::denorm\_min \(\) [inline], [static], [constexpr],  
[noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### **epsilon()**

```
static constexpr int std::numeric\_limits< int >::epsilon \(\) [inline], [static], [constexpr],  
[noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

**infinity()**

```
static constexpr int std::numeric_limits< int >::infinity () [inline], [static], [constexpr],
[noexcept]
```

The representation of positive infinity, if has\_infinity.

**lowest()**

```
static constexpr int std::numeric_limits< int >::lowest () [inline], [static], [constexpr],
[noexcept]
```

A finite value x such that there is no other finite value y where  $y < x$ .

**max()**

```
static constexpr int std::numeric_limits< int >::max () [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

**min()**

```
static constexpr int std::numeric_limits< int >::min () [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

```
static constexpr int std::numeric_limits< int >::quiet_NaN () [inline], [static], [constexpr],
[noexcept]
```

The representation of a quiet Not a Number, if has\_quiet\_NaN.

**round\_error()**

```
static constexpr int std::numeric_limits< int >::round_error () [inline], [static], [constexpr],
[noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr int std::numeric_limits< int >::signaling_NaN () [inline], [static], [constexpr],
[noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.

**5.719.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See std::float\_denorm\_style for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

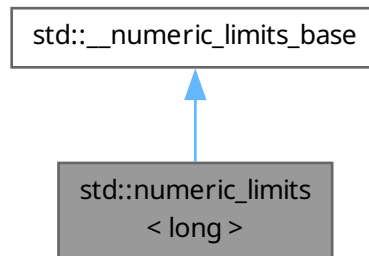
- [limits](#)



## 5.720 std::numeric\_limits< long > Struct Reference

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< long >:



### Static Public Member Functions

- static constexpr long [denorm\\_min](#) () noexcept
- static constexpr long **denorm\_min** () noexcept
- static constexpr long [epsilon](#) () noexcept
- static constexpr long **epsilon** () noexcept
- static constexpr long [infinity](#) () noexcept
- static constexpr long **infinity** () noexcept
- static constexpr long [lowest](#) () noexcept
- static constexpr long **lowest** () noexcept
- static constexpr long [max](#) () noexcept
- static constexpr long **max** () noexcept
- static constexpr long [min](#) () noexcept
- static constexpr long **min** () noexcept
- static constexpr long [quiet\\_NaN](#) () noexcept
- static constexpr long **quiet\_NaN** () noexcept
- static constexpr long [round\\_error](#) () noexcept
- static constexpr long **round\_error** () noexcept
- static constexpr long [signaling\\_NaN](#) () noexcept
- static constexpr long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**

- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.720.1 Detailed Description

`numeric_limits<long>` specialization.

### 5.720.2 Member Function Documentation

#### **denorm\_min()**

```
static constexpr long std::numeric_limits< long >::denorm_min () [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

**epsilon()**

```
static constexpr long std::numeric_limits< long >::epsilon () [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

**infinity()**

```
static constexpr long std::numeric_limits< long >::infinity () [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

**lowest()**

```
static constexpr long std::numeric_limits< long >::lowest () [inline], [static], [constexpr], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

**max()**

```
static constexpr long std::numeric_limits< long >::max () [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

**min()**

```
static constexpr long std::numeric_limits< long >::min () [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

```
static constexpr long std::numeric_limits< long >::quiet_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

**round\_error()**

```
static constexpr long std::numeric_limits< long >::round_error () [inline], [static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr long std::numeric_limits< long >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

**5.720.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.721 std::numeric\_limits< long double > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< long double >:

**Static Public Member Functions**

- static constexpr long double [denorm\\_min](#) () noexcept
- static constexpr long double **denorm\_min** () noexcept
- static constexpr long double [epsilon](#) () noexcept
- static constexpr long double **epsilon** () noexcept
- static constexpr long double [infinity](#) () noexcept
- static constexpr long double **infinity** () noexcept
- static constexpr long double [lowest](#) () noexcept
- static constexpr long double **lowest** () noexcept
- static constexpr long double [max](#) () noexcept
- static constexpr long double **max** () noexcept
- static constexpr long double [min](#) () noexcept
- static constexpr long double **min** () noexcept

- static constexpr long double [quiet\\_NaN](#) () noexcept
- static constexpr long double **quiet\_NaN** () noexcept
- static constexpr long double [round\\_error](#) () noexcept
- static constexpr long double **round\_error** () noexcept
- static constexpr long double [signaling\\_NaN](#) () noexcept
- static constexpr long double **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**

- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **`tinyness_before`**
- static constexpr bool [traps](#)
- static constexpr bool **`traps`**

### 5.721.1 Detailed Description

`numeric_limits<long double>` specialization.

### 5.721.2 Member Function Documentation

#### **`denorm_min()`**

```
static constexpr long double std::numeric\_limits< long double >::denorm\_min \(\) [inline], [static],  
[constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### **`epsilon()`**

```
static constexpr long double std::numeric\_limits< long double >::epsilon \(\) [inline], [static],  
[constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### **`infinity()`**

```
static constexpr long double std::numeric\_limits< long double >::infinity \(\) [inline], [static],  
[constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

#### **`lowest()`**

```
static constexpr long double std::numeric\_limits< long double >::lowest \(\) [inline], [static],  
[constexpr], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

#### **`max()`**

```
static constexpr long double std::numeric\_limits< long double >::max \(\) [inline], [static],  
[constexpr], [noexcept]
```

The maximum finite value.

#### **`min()`**

```
static constexpr long double std::numeric\_limits< long double >::min \(\) [inline], [static],  
[constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

#### **`quiet_NaN()`**

```
static constexpr long double std::numeric\_limits< long double >::quiet\_NaN \(\) [inline], [static],  
[constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.



**round\_error()**

```
static constexpr long double std::numeric_limits< long double >::round_error () [inline], [static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr long double std::numeric_limits< long double >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

**5.721.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.722 std::numeric\_limits< long long > Struct Reference**

```
#include <limits>
```

Inheritance diagram for `std::numeric_limits< long long >`:



**Static Public Member Functions**

- static constexpr long long [denorm\\_min](#) () noexcept
- static constexpr long long **denorm\_min** () noexcept
- static constexpr long long [epsilon](#) () noexcept
- static constexpr long long **epsilon** () noexcept
- static constexpr long long [infinity](#) () noexcept
- static constexpr long long **infinity** () noexcept
- static constexpr long long [lowest](#) () noexcept
- static constexpr long long **lowest** () noexcept
- static constexpr long long [max](#) () noexcept
- static constexpr long long **max** () noexcept
- static constexpr long long [min](#) () noexcept
- static constexpr long long **min** () noexcept
- static constexpr long long [quiet\\_NaN](#) () noexcept
- static constexpr long long **quiet\_NaN** () noexcept
- static constexpr long long [round\\_error](#) () noexcept
- static constexpr long long **round\_error** () noexcept
- static constexpr long long [signaling\\_NaN](#) () noexcept
- static constexpr long long **signaling\_NaN** () noexcept

**Static Public Attributes**

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)

- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

#### 5.722.1 Detailed Description

`numeric_limits<long long>` specialization.

#### 5.722.2 Member Function Documentation

##### **denorm\_min()**

```
static constexpr long long std::numeric\_limits< long long >::denorm\_min \(\) [inline], [static],  
[constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

##### **epsilon()**

```
static constexpr long long std::numeric\_limits< long long >::epsilon \(\) [inline], [static],  
[constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

##### **infinity()**

```
static constexpr long long std::numeric\_limits< long long >::infinity \(\) [inline], [static],  
[constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

##### **lowest()**

```
static constexpr long long std::numeric\_limits< long long >::lowest \(\) [inline], [static], [constexpr],  
[noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

##### **max()**

```
static constexpr long long std::numeric\_limits< long long >::max \(\) [inline], [static], [constexpr],  
[noexcept]
```

The maximum finite value.

**min()**

```
static constexpr long long std::numeric_limits< long long >::min () [inline], [static], [constexpr],
[noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

```
static constexpr long long std::numeric_limits< long long >::quiet_NaN () [inline], [static],
[constexpr], [noexcept]
```

The representation of a quiet Not a Number, if has\_quiet\_NaN.

**round\_error()**

```
static constexpr long long std::numeric_limits< long long >::round_error () [inline], [static],
[constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr long long std::numeric_limits< long long >::signaling_NaN () [inline], [static],
[constexpr], [noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.

**5.722.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See std::float\_denorm\_style for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`  
True if the type has a representation for a signaling Not a Number.

**is\_bounded**

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`  
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`  
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`  
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range `[min(),max())`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

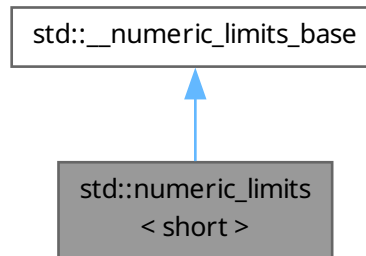
- [limits](#)

**5.723 std::numeric\_limits< short > Struct Reference**

```
#include <limits>
```



Inheritance diagram for `std::numeric_limits< short >`:



### Static Public Member Functions

- static constexpr short `denorm_min` () noexcept
- static constexpr short `denorm_min` () noexcept
- static constexpr short `epsilon` () noexcept
- static constexpr short `epsilon` () noexcept
- static constexpr short `infinity` () noexcept
- static constexpr short `infinity` () noexcept
- static constexpr short `lowest` () noexcept
- static constexpr short `lowest` () noexcept
- static constexpr short `max` () noexcept
- static constexpr short `max` () noexcept
- static constexpr short `min` () noexcept
- static constexpr short `min` () noexcept
- static constexpr short `quiet_NaN` () noexcept
- static constexpr short `quiet_NaN` () noexcept
- static constexpr short `round_error` () noexcept
- static constexpr short `round_error` () noexcept
- static constexpr short `signaling_NaN` () noexcept
- static constexpr short `signaling_NaN` () noexcept

### Static Public Attributes

- static constexpr int `digits`
- static constexpr int `digits`
- static constexpr int `digits10`
- static constexpr int `digits10`
- static constexpr `float_denorm_style` `has_denorm`
- static constexpr `float_denorm_style` `has_denorm`
- static constexpr bool `has_denorm_loss`
- static constexpr bool `has_denorm_loss`
- static constexpr bool `has_infinity`
- static constexpr bool `has_infinity`
- static constexpr bool `has_quiet_NaN`

- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.723.1 Detailed Description

`numeric_limits<short>` specialization.

### 5.723.2 Member Function Documentation

#### **denorm\_min()**

```
static constexpr short std::numeric_limits< short >::denorm_min () [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### **epsilon()**

```
static constexpr short std::numeric_limits< short >::epsilon () [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

**infinity()**

```
static constexpr short std::numeric_limits< short >::infinity () [inline], [static], [constexpr],  
[noexcept]
```

The representation of positive infinity, if has\_infinity.

**lowest()**

```
static constexpr short std::numeric_limits< short >::lowest () [inline], [static], [constexpr],  
[noexcept]
```

A finite value x such that there is no other finite value y where  $y < x$ .

**max()**

```
static constexpr short std::numeric_limits< short >::max () [inline], [static], [constexpr],  
[noexcept]
```

The maximum finite value.

**min()**

```
static constexpr short std::numeric_limits< short >::min () [inline], [static], [constexpr],  
[noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

```
static constexpr short std::numeric_limits< short >::quiet_NaN () [inline], [static], [constexpr],  
[noexcept]
```

The representation of a quiet Not a Number, if has\_quiet\_NaN.

**round\_error()**

```
static constexpr short std::numeric_limits< short >::round_error () [inline], [static], [constexpr],  
[noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr short std::numeric_limits< short >::signaling_NaN () [inline], [static], [constexpr],  
[noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.

### 5.723.3 Member Data Documentation

**digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See std::float\_denorm\_style for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`  
See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`  
True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

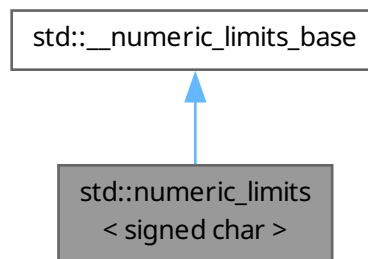
The documentation for this struct was generated from the following file:

- [limits](#)

**5.724 std::numeric\_limits< signed char > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< signed char >:

**Static Public Member Functions**

- static constexpr signed char [denorm\\_min](#) () noexcept
- static constexpr signed char **denorm\_min** () noexcept
- static constexpr signed char [epsilon](#) () noexcept
- static constexpr signed char **epsilon** () noexcept
- static constexpr signed char [infinity](#) () noexcept
- static constexpr signed char **infinity** () noexcept
- static constexpr signed char [lowest](#) () noexcept
- static constexpr signed char **lowest** () noexcept
- static constexpr signed char [max](#) () noexcept
- static constexpr signed char **max** () noexcept
- static constexpr signed char [min](#) () noexcept
- static constexpr signed char **min** () noexcept
- static constexpr signed char [quiet\\_NaN](#) () noexcept
- static constexpr signed char **quiet\_NaN** () noexcept
- static constexpr signed char [round\\_error](#) () noexcept
- static constexpr signed char **round\_error** () noexcept
- static constexpr signed char [signaling\\_NaN](#) () noexcept
- static constexpr signed char **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.724.1 Detailed Description

numeric\_limits<signed char> specialization.

### 5.724.2 Member Function Documentation

#### denorm\_min()

```
static constexpr signed char std::numeric_limits< signed char >::denorm_min () [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### epsilon()

```
static constexpr signed char std::numeric_limits< signed char >::epsilon () [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### infinity()

```
static constexpr signed char std::numeric_limits< signed char >::infinity () [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

#### lowest()

```
static constexpr signed char std::numeric_limits< signed char >::lowest () [inline], [static], [constexpr], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

#### max()

```
static constexpr signed char std::numeric_limits< signed char >::max () [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

#### min()

```
static constexpr signed char std::numeric_limits< signed char >::min () [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

#### quiet\_NaN()

```
static constexpr signed char std::numeric_limits< signed char >::quiet_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

#### round\_error()

```
static constexpr signed char std::numeric_limits< signed char >::round_error () [inline], [static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

#### signaling\_NaN()

```
static constexpr signed char std::numeric_limits< signed char >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.



### 5.724.3 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that *radix* raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that *radix* raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.725 std::numeric\_limits< unsigned char > Struct Reference**

```
#include <limits>
```

Inheritance diagram for `std::numeric_limits< unsigned char >`:

**Static Public Member Functions**

- static constexpr unsigned char [denorm\\_min](#) () noexcept
- static constexpr unsigned char **denorm\_min** () noexcept
- static constexpr unsigned char [epsilon](#) () noexcept
- static constexpr unsigned char **epsilon** () noexcept
- static constexpr unsigned char [infinity](#) () noexcept
- static constexpr unsigned char **infinity** () noexcept
- static constexpr unsigned char [lowest](#) () noexcept
- static constexpr unsigned char **lowest** () noexcept
- static constexpr unsigned char [max](#) () noexcept
- static constexpr unsigned char **max** () noexcept
- static constexpr unsigned char [min](#) () noexcept
- static constexpr unsigned char **min** () noexcept

- static constexpr unsigned char [quiet\\_NaN](#) () noexcept
- static constexpr unsigned char **quiet\_NaN** () noexcept
- static constexpr unsigned char [round\\_error](#) () noexcept
- static constexpr unsigned char **round\_error** () noexcept
- static constexpr unsigned char [signaling\\_NaN](#) () noexcept
- static constexpr unsigned char **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**

- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.725.1 Detailed Description

numeric\_limits<unsigned char> specialization.

### 5.725.2 Member Function Documentation

#### denorm\_min()

```
static constexpr unsigned char std::numeric\_limits< unsigned char >::denorm_min () [inline],  
[static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### epsilon()

```
static constexpr unsigned char std::numeric\_limits< unsigned char >::epsilon () [inline], [static],  
[constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### infinity()

```
static constexpr unsigned char std::numeric\_limits< unsigned char >::infinity () [inline], [static],  
[constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

#### lowest()

```
static constexpr unsigned char std::numeric\_limits< unsigned char >::lowest () [inline], [static],  
[constexpr], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

#### max()

```
static constexpr unsigned char std::numeric\_limits< unsigned char >::max () [inline], [static],  
[constexpr], [noexcept]
```

The maximum finite value.

#### min()

```
static constexpr unsigned char std::numeric\_limits< unsigned char >::min () [inline], [static],  
[constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

#### quiet\_NaN()

```
static constexpr unsigned char std::numeric\_limits< unsigned char >::quiet_NaN () [inline],  
[static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

**round\_error()**

```
static constexpr unsigned char std::numeric_limits< unsigned char >::round_error () [inline],  
[static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr unsigned char std::numeric_limits< unsigned char >::signaling_NaN () [inline],  
[static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.

**5.725.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that radix raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.726 std::numeric\_limits< unsigned int > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< unsigned int >:





### Static Public Member Functions

- static constexpr unsigned int [denorm\\_min](#) () noexcept
- static constexpr unsigned int **denorm\_min** () noexcept
- static constexpr unsigned int [epsilon](#) () noexcept
- static constexpr unsigned int **epsilon** () noexcept
- static constexpr unsigned int [infinity](#) () noexcept
- static constexpr unsigned int **infinity** () noexcept
- static constexpr unsigned int [lowest](#) () noexcept
- static constexpr unsigned int **lowest** () noexcept
- static constexpr unsigned int [max](#) () noexcept
- static constexpr unsigned int **max** () noexcept
- static constexpr unsigned int [min](#) () noexcept
- static constexpr unsigned int **min** () noexcept
- static constexpr unsigned int [quiet\\_NaN](#) () noexcept
- static constexpr unsigned int **quiet\_NaN** () noexcept
- static constexpr unsigned int [round\\_error](#) () noexcept
- static constexpr unsigned int **round\_error** () noexcept
- static constexpr unsigned int [signaling\\_NaN](#) () noexcept
- static constexpr unsigned int **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)

- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.726.1 Detailed Description

numeric\_limits<unsigned int> specialization.

### 5.726.2 Member Function Documentation

#### denorm\_min()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::denorm_min () [inline], [static],
[constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### epsilon()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::epsilon () [inline], [static],
[constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### infinity()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::infinity () [inline], [static],
[constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

#### lowest()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::lowest () [inline], [static],
[constexpr], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

#### max()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::max () [inline], [static],
[constexpr], [noexcept]
```

The maximum finite value.

**min()**

```
static constexpr unsigned int std::numeric_limits< unsigned int >::min () [inline], [static],  
[constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

```
static constexpr unsigned int std::numeric_limits< unsigned int >::quiet_NaN () [inline], [static],  
[constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

**round\_error()**

```
static constexpr unsigned int std::numeric_limits< unsigned int >::round_error () [inline],  
[static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr unsigned int std::numeric_limits< unsigned int >::signaling_NaN () [inline],  
[static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

**5.726.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()), the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.727 std::numeric\_limits< unsigned long > Struct Reference**

```
#include <limits>
```

Inheritance diagram for `std::numeric_limits< unsigned long >`:



### Static Public Member Functions

- static constexpr unsigned long `denorm_min` () noexcept
- static constexpr unsigned long `denorm_min` () noexcept
- static constexpr unsigned long `epsilon` () noexcept
- static constexpr unsigned long `epsilon` () noexcept
- static constexpr unsigned long `infinity` () noexcept
- static constexpr unsigned long `infinity` () noexcept
- static constexpr unsigned long `lowest` () noexcept
- static constexpr unsigned long `lowest` () noexcept
- static constexpr unsigned long `max` () noexcept
- static constexpr unsigned long `max` () noexcept
- static constexpr unsigned long `min` () noexcept
- static constexpr unsigned long `min` () noexcept
- static constexpr unsigned long `quiet_NaN` () noexcept
- static constexpr unsigned long `quiet_NaN` () noexcept
- static constexpr unsigned long `round_error` () noexcept
- static constexpr unsigned long `round_error` () noexcept
- static constexpr unsigned long `signaling_NaN` () noexcept
- static constexpr unsigned long `signaling_NaN` () noexcept

### Static Public Attributes

- static constexpr int `digits`
- static constexpr int `digits`
- static constexpr int `digits10`
- static constexpr int `digits10`
- static constexpr `float_denorm_style` `has_denorm`
- static constexpr `float_denorm_style` `has_denorm`
- static constexpr bool `has_denorm_loss`
- static constexpr bool `has_denorm_loss`
- static constexpr bool `has_infinity`
- static constexpr bool `has_infinity`
- static constexpr bool `has_quiet_NaN`

- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.727.1 Detailed Description

numeric\_limits<unsigned long> specialization.

### 5.727.2 Member Function Documentation

#### denorm\_min()

```
static constexpr unsigned long std::numeric\_limits< unsigned long >::denorm_min () [inline],
[static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### epsilon()

```
static constexpr unsigned long std::numeric\_limits< unsigned long >::epsilon () [inline], [static],
[constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

**infinity()**

```
static constexpr unsigned long std::numeric_limits< unsigned long >::infinity () [inline], [static],
[constexpr], [noexcept]
```

The representation of positive infinity, if has\_infinity.

**lowest()**

```
static constexpr unsigned long std::numeric_limits< unsigned long >::lowest () [inline], [static],
[constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where y < x.

**max()**

```
static constexpr unsigned long std::numeric_limits< unsigned long >::max () [inline], [static],
[constexpr], [noexcept]
```

The maximum finite value.

**min()**

```
static constexpr unsigned long std::numeric_limits< unsigned long >::min () [inline], [static],
[constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

```
static constexpr unsigned long std::numeric_limits< unsigned long >::quiet_NaN () [inline],
[static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if has\_quiet\_NaN.

**round\_error()**

```
static constexpr unsigned long std::numeric_limits< unsigned long >::round_error () [inline],
[static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr unsigned long std::numeric_limits< unsigned long >::signaling_NaN () [inline],
[static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.

**5.727.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.



**has\_denorm**

`float_denorm_style` `std::__numeric_limits_base::has_denorm` [static], [constexpr]  
See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

`bool` `std::__numeric_limits_base::has_denorm_loss` [static], [constexpr]  
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

`bool` `std::__numeric_limits_base::has_infinity` [static], [constexpr]  
True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

`bool` `std::__numeric_limits_base::has_quiet_NaN` [static], [constexpr]  
True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

`bool` `std::__numeric_limits_base::has_signaling_NaN` [static], [constexpr]  
True if the type has a representation for a signaling Not a Number.

**is\_bounded**

`bool` `std::__numeric_limits_base::is_bounded` [static], [constexpr]  
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

`bool` `std::__numeric_limits_base::is_exact` [static], [constexpr]  
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

`bool` `std::__numeric_limits_base::is_iec559` [static], [constexpr]  
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool` `std::__numeric_limits_base::is_integer` [static], [constexpr]  
True if the type is integer.

**is\_modulo**

`bool` `std::__numeric_limits_base::is_modulo` [static], [constexpr]  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range `[min(),max()]`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.728 std::numeric\_limits< unsigned long long > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< unsigned long long >:

**Static Public Member Functions**

- static constexpr unsigned long long [denorm\\_min](#) () noexcept
- static constexpr unsigned long long **denorm\_min** () noexcept
- static constexpr unsigned long long [epsilon](#) () noexcept
- static constexpr unsigned long long **epsilon** () noexcept
- static constexpr unsigned long long [infinity](#) () noexcept
- static constexpr unsigned long long **infinity** () noexcept
- static constexpr unsigned long long [lowest](#) () noexcept
- static constexpr unsigned long long **lowest** () noexcept
- static constexpr unsigned long long [max](#) () noexcept
- static constexpr unsigned long long **max** () noexcept
- static constexpr unsigned long long [min](#) () noexcept
- static constexpr unsigned long long **min** () noexcept
- static constexpr unsigned long long [quiet\\_NaN](#) () noexcept
- static constexpr unsigned long long **quiet\_NaN** () noexcept
- static constexpr unsigned long long [round\\_error](#) () noexcept
- static constexpr unsigned long long **round\_error** () noexcept
- static constexpr unsigned long long [signaling\\_NaN](#) () noexcept
- static constexpr unsigned long long **signaling\_NaN** () noexcept

**Static Public Attributes**

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

**5.728.1 Detailed Description**

numeric\_limits<unsigned long long> specialization.

## 5.728.2 Member Function Documentation

### denorm\_min()

```
static constexpr unsigned long long std::numeric_limits< unsigned long long >::denorm_min ()  
[inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

### epsilon()

```
static constexpr unsigned long long std::numeric_limits< unsigned long long >::epsilon () [inline],  
[static], [constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

### infinity()

```
static constexpr unsigned long long std::numeric_limits< unsigned long long >::infinity () [inline],  
[static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

### lowest()

```
static constexpr unsigned long long std::numeric_limits< unsigned long long >::lowest () [inline],  
[static], [constexpr], [noexcept]
```

A finite value  $x$  such that there is no other finite value  $y$  where  $y < x$ .

### max()

```
static constexpr unsigned long long std::numeric_limits< unsigned long long >::max () [inline],  
[static], [constexpr], [noexcept]
```

The maximum finite value.

### min()

```
static constexpr unsigned long long std::numeric_limits< unsigned long long >::min () [inline],  
[static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

### quiet\_NaN()

```
static constexpr unsigned long long std::numeric_limits< unsigned long long >::quiet_NaN () [inline],  
[static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

### round\_error()

```
static constexpr unsigned long long std::numeric_limits< unsigned long long >::round_error ()  
[inline], [static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

### signaling\_NaN()

```
static constexpr unsigned long long std::numeric_limits< unsigned long long >::signaling_NaN ()  
[inline], [static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

### 5.728.3 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that *radix* raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that *radix* raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.729 std::numeric\_limits< unsigned short > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< unsigned short >:

**Static Public Member Functions**

- static constexpr unsigned short [denorm\\_min](#) () noexcept
- static constexpr unsigned short **denorm\_min** () noexcept
- static constexpr unsigned short [epsilon](#) () noexcept
- static constexpr unsigned short **epsilon** () noexcept
- static constexpr unsigned short [infinity](#) () noexcept
- static constexpr unsigned short **infinity** () noexcept
- static constexpr unsigned short [lowest](#) () noexcept
- static constexpr unsigned short **lowest** () noexcept
- static constexpr unsigned short [max](#) () noexcept
- static constexpr unsigned short **max** () noexcept
- static constexpr unsigned short [min](#) () noexcept
- static constexpr unsigned short **min** () noexcept



- static constexpr unsigned short [quiet\\_NaN](#) () noexcept
- static constexpr unsigned short **quiet\_NaN** () noexcept
- static constexpr unsigned short [round\\_error](#) () noexcept
- static constexpr unsigned short **round\_error** () noexcept
- static constexpr unsigned short [signaling\\_NaN](#) () noexcept
- static constexpr unsigned short **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**

- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.729.1 Detailed Description

numeric\_limits<unsigned short> specialization.

### 5.729.2 Member Function Documentation

#### denorm\_min()

```
static constexpr unsigned short std::numeric_limits< unsigned short >::denorm_min () [inline],  
[static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where has\_denorm is false, this is the minimum positive normalized value.

#### epsilon()

```
static constexpr unsigned short std::numeric_limits< unsigned short >::epsilon () [inline],  
[static], [constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### infinity()

```
static constexpr unsigned short std::numeric_limits< unsigned short >::infinity () [inline],  
[static], [constexpr], [noexcept]
```

The representation of positive infinity, if has\_infinity.

#### lowest()

```
static constexpr unsigned short std::numeric_limits< unsigned short >::lowest () [inline], [static],  
[constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where  $y < x$ .

#### max()

```
static constexpr unsigned short std::numeric_limits< unsigned short >::max () [inline], [static],  
[constexpr], [noexcept]
```

The maximum finite value.

#### min()

```
static constexpr unsigned short std::numeric_limits< unsigned short >::min () [inline], [static],  
[constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

#### quiet\_NaN()

```
static constexpr unsigned short std::numeric_limits< unsigned short >::quiet_NaN () [inline],  
[static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if has\_quiet\_NaN.

**round\_error()**

```
static constexpr unsigned short std::numeric_limits< unsigned short >::round_error () [inline],  
[static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr unsigned short std::numeric_limits< unsigned short >::signaling_NaN () [inline],  
[static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

**5.729.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving `+`, `-`, or `*` on values of that type whose result would fall outside the range `[min(),max())`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.730 std::numeric\_limits< wchar\_t > Struct Reference**

```
#include <limits>
```

Inheritance diagram for `std::numeric_limits< wchar_t >`:



### Static Public Member Functions

- static constexpr wchar\_t [denorm\\_min](#) () noexcept
- static constexpr wchar\_t **denorm\_min** () noexcept
- static constexpr wchar\_t [epsilon](#) () noexcept
- static constexpr wchar\_t **epsilon** () noexcept
- static constexpr wchar\_t [infinity](#) () noexcept
- static constexpr wchar\_t **infinity** () noexcept
- static constexpr wchar\_t [lowest](#) () noexcept
- static constexpr wchar\_t **lowest** () noexcept
- static constexpr wchar\_t [max](#) () noexcept
- static constexpr wchar\_t **max** () noexcept
- static constexpr wchar\_t [min](#) () noexcept
- static constexpr wchar\_t **min** () noexcept
- static constexpr wchar\_t [quiet\\_NaN](#) () noexcept
- static constexpr wchar\_t **quiet\_NaN** () noexcept
- static constexpr wchar\_t [round\\_error](#) () noexcept
- static constexpr wchar\_t **round\_error** () noexcept
- static constexpr wchar\_t [signaling\\_NaN](#) () noexcept
- static constexpr wchar\_t **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int **digits**
- static constexpr int [digits10](#)
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_specialized](#)
- static constexpr bool **is\_specialized**
- static constexpr int [max\\_digits10](#)

- static constexpr int **max\_digits10**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **max\_exponent10**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **min\_exponent10**
- static constexpr int [radix](#)
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [traps](#)
- static constexpr bool **traps**

### 5.730.1 Detailed Description

`numeric_limits<wchar_t>` specialization.

### 5.730.2 Member Function Documentation

#### **denorm\_min()**

```
static constexpr wchar_t std::numeric\_limits< wchar\_t >::denorm\_min \(\) [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

#### **epsilon()**

```
static constexpr wchar_t std::numeric\_limits< wchar\_t >::epsilon \(\) [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

#### **infinity()**

```
static constexpr wchar_t std::numeric\_limits< wchar\_t >::infinity \(\) [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

#### **lowest()**

```
static constexpr wchar_t std::numeric\_limits< wchar\_t >::lowest \(\) [inline], [static], [constexpr], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

#### **max()**

```
static constexpr wchar_t std::numeric\_limits< wchar\_t >::max \(\) [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

**min()**

```
static constexpr wchar_t std::numeric_limits< wchar_t >::min () [inline], [static], [constexpr],
[noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

**quiet\_NaN()**

```
static constexpr wchar_t std::numeric_limits< wchar_t >::quiet_NaN () [inline], [static], [constexpr],
[noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

**round\_error()**

```
static constexpr wchar_t std::numeric_limits< wchar_t >::round_error () [inline], [static],
[constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
static constexpr wchar_t std::numeric_limits< wchar_t >::signaling_NaN () [inline], [static],
[constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

**5.730.3 Member Data Documentation****digits**

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

**digits10**

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

**has\_denorm**

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.



**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()), the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.

**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.731 std::numeric\_limits<\_CharT> Class Template Reference**

```
#include <locale_facets.h>
```

Inheritance diagram for `std::numpunct<_CharT>`:



### Public Types

- typedef `__numpunct_cache<_CharT>` `__cache_type`
- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

### Public Member Functions

- `numpunct` (`__c_locale __cloc`, `size_t __refs=0`)
- `numpunct` (`__cache_type *__cache`, `size_t __refs=0`)
- `numpunct` (`size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual `~numpunct` ()
- void `_M_initialize_numpunct` (`__c_locale __cloc`)
- void `_M_initialize_numpunct` (`__c_locale __cloc`)
- void `_M_initialize_numpunct` (`__c_locale __cloc=0`)
- virtual `char_type do_decimal_point` () const
- virtual `string_type do_falsename` () const

- virtual [string do\\_grouping](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const
- virtual [string\\_type do\\_truename](#) () const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Protected Attributes

- `__cache_type * _M_data`

#### 5.731.1 Detailed Description

**template<typename \_CharT>**  
**class std::numpunct<\_CharT>**

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a numpunct facet.

#### 5.731.2 Member Typedef Documentation

##### char\_type

```
template<typename _CharT>
typedef _CharT std::numpunct<_CharT>::char_type
```

Public typedefs.

##### string\_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::numpunct<_CharT>::string_type
```

Public typedefs.

#### 5.731.3 Constructor & Destructor Documentation

##### numpunct() [1/3]

```
template<typename _CharT>
std::numpunct<_CharT>::numpunct (
    size_t __refs = 0) [inline], [explicit]
```

Numpunct constructor.

##### Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__refs</code> | Refcount to pass to the base class. |
|---------------------|-------------------------------------|

References [std::locale::facet::facet\(\)](#).

### numpunct() [2/3]

```
template<typename _CharT>
std::numpunct< _CharT >::numpunct (
    __cache_type * __cache,
    size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

#### Parameters

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__cache</code> | __numpunct_cache object.            |
| <code>__refs</code>  | Refcount to pass to the base class. |

References [std::locale::facet::facet\(\)](#).

### numpunct() [3/3]

```
template<typename _CharT>
std::numpunct< _CharT >::numpunct (
    __c_locale __cloc,
    size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

#### Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__cloc</code> | The C locale.                       |
| <code>__refs</code> | Refcount to pass to the base class. |

References [std::locale::facet::facet\(\)](#).

### ~numpunct()

```
template<typename _CharT>
virtual std::numpunct< _CharT >::~~numpunct () [protected], [virtual]
```

Destructor.

## 5.731.4 Member Function Documentation

### decimal\_point()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::decimal_point () const [inline]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `returning numpunct<char_type>::do_decimal_point()`.

#### Returns

*char\_type* representing a decimal point.

References [do\\_decimal\\_point\(\)](#).

**do\_decimal\_point()**

```
template<typename _CharT>
virtual char\_type std::num_punct<_CharT>::do_decimal_point () const [inline], [protected], [virtual]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Referenced by [decimal\\_point\(\)](#).

**do\_falsename()**

```
template<typename _CharT>
virtual string\_type std::num_punct<_CharT>::do_falsename () const [inline], [protected], [virtual]
```

Return string representation of bool false.

Returns a `string_type` containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of false.

Referenced by [falsename\(\)](#).

**do\_grouping()**

```
template<typename _CharT>
virtual string std::num_punct<_CharT>::do_grouping () const [inline], [protected], [virtual]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Referenced by [grouping\(\)](#).

**do\_thousands\_sep()**

```
template<typename _CharT>
virtual char\_type std::num_punct<_CharT>::do_thousands_sep () const [inline], [protected], [virtual]
```

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Referenced by [thousands\\_sep\(\)](#).

**do\_truename()**

```
template<typename _CharT>
```

```
virtual string_type std::num_punct< _CharT >::do_truename () const [inline], [protected], [virtual]
```

Return string representation of bool true.

Returns a string\_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

string\_type representing printed form of true.

Referenced by [truename\(\)](#).

**falsename()**

```
template<typename _CharT>
```

```
string_type std::num_punct< _CharT >::false_name () const [inline]
```

Return string representation of bool false.

This function returns a string\_type containing the text representation for false bool variables. It does so by calling num\_punct<char\_type>::do\_false\_name().

**Returns**

string\_type representing printed form of false.

References [do\\_false\\_name\(\)](#).

**grouping()**

```
template<typename _CharT>
```

```
string std::num_punct< _CharT >::grouping () const [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling num\_punct<char\_type>::do\_grouping().

**Returns**

string representing grouping specification.

References [do\\_grouping\(\)](#).

**thousands\_sep()**

```
template<typename _CharT>
```

```
char_type std::num_punct< _CharT >::thousands_sep () const [inline]
```

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning num\_punct<char\_type>::do\_thousands\_sep().

**Returns**

char\_type representing a thousands separator.

References [do\\_thousands\\_sep\(\)](#).

**truename()**

```
template<typename _CharT>
string_type std::numpunct<_CharT>::true_name () const [inline]
```

Return string representation of bool true.

This function returns a string\_type containing the text representation for true bool variables. It does so by calling numpunct<char\_type>::do\_true\_name().

**Returns**

string\_type representing printed form of true.

References [do\\_true\\_name\(\)](#).

**5.731.5 Member Data Documentation****id**

```
template<typename _CharT>
locale::id std::numpunct<_CharT>::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

**5.732 std::numpunct\_byname<\_CharT> Class Template Reference**

```
#include <locale_facets.h>
```

Inheritance diagram for std::numpunct\_byname<\_CharT>:

**Public Types**

- typedef \_\_numpunct\_cache<\_CharT> \_\_cache\_type
- typedef \_CharT char\_type
- typedef [basic\\_string](#)<\_CharT> string\_type



### Public Member Functions

- **numpunct\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **numpunct\_byname** (const [string](#) & \_\_s, size\_t \_\_refs=0)
- char\_type **decimal\_point** () const
- [string\\_type](#) **falsename** () const
- [string](#) **grouping** () const
- char\_type **thousands\_sep** () const
- [string\\_type](#) **truename** () const

### Static Public Attributes

- static [locale::id](#) **id**

### Protected Member Functions

- void **\_M\_initialize\_numpunct** (\_\_c\_locale \_\_cloc)
- void **\_M\_initialize\_numpunct** (\_\_c\_locale \_\_cloc)
- void **\_M\_initialize\_numpunct** (\_\_c\_locale \_\_cloc=0)
- virtual char\_type **do\_decimal\_point** () const
- virtual [string\\_type](#) **do\_falsename** () const
- virtual [string](#) **do\_grouping** () const
- virtual char\_type **do\_thousands\_sep** () const
- virtual [string\\_type](#) **do\_truename** () const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale & \_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_type\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_cache\_type \* **\_M\_data**

#### 5.732.1 Detailed Description

```
template<typename _CharT>
class std::numpunct_byname< _CharT >
```

class numpunct\_byname [22.2.3.2].

#### 5.732.2 Member Function Documentation

##### decimal\_point()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::decimal_point () const [inline], [inherited]
```

Return decimal point character.

This function returns a char\_type to use as a decimal point. It does so by returning `numpunct<char_type>::do_decimal_point()`.

**Returns**

*char\_type* representing a decimal point.

References [do\\_decimal\\_point\(\)](#).

**do\_decimal\_point()**

```
template<typename _CharT>
virtual char_type std::num_punct<_CharT>::do_decimal_point () const [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.

Returns a *char\_type* to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Referenced by [decimal\\_point\(\)](#).

**do\_falsename()**

```
template<typename _CharT>
virtual string_type std::num_punct<_CharT>::do_falsename () const [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool false.

Returns a *string\_type* containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of false.

Referenced by [falsename\(\)](#).

**do\_grouping()**

```
template<typename _CharT>
virtual string std::num_punct<_CharT>::do_grouping () const [inline], [protected], [virtual],
[inherited]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Referenced by [grouping\(\)](#).

**do\_thousands\_sep()**

```
template<typename _CharT>
virtual char_type std::num_punct<_CharT>::do_thousands_sep () const [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.

Returns a *char\_type* to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Referenced by [thousands\\_sep\(\)](#).

**do\_truename()**

```
template<typename _CharT>
virtual string_type std::num_punct< _CharT >::do_truename () const [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool true.

Returns a *string\_type* containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of true.

Referenced by [truename\(\)](#).

**falsename()**

```
template<typename _CharT>
string_type std::num_punct< _CharT >::falsename () const [inline], [inherited]
```

Return string representation of bool false.

This function returns a *string\_type* containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

**Returns**

*string\_type* representing printed form of false.

References [do\\_falsename\(\)](#).

**grouping()**

```
template<typename _CharT>
string std::num_punct< _CharT >::grouping () const [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `num_punct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

References [do\\_grouping\(\)](#).

**thousands\_sep()**

```
template<typename _CharT>
```

```
char_type std::numpunct< _CharT >::thousands_sep () const [inline], [inherited]
```

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning returning numpunct<char↵\_type>::do\_thousands\_sep().

**Returns**

char\_type representing a thousands separator.

References [do\\_thousands\\_sep\(\)](#).

**truename()**

```
template<typename _CharT>
```

```
string_type std::numpunct< _CharT >::true_name () const [inline], [inherited]
```

Return string representation of bool true.

This function returns a string\_type containing the text representation for true bool variables. It does so by calling numpunct<char\_type>::do\_true\_name().

**Returns**

string\_type representing printed form of true.

References [do\\_true\\_name\(\)](#).

**5.732.3 Member Data Documentation****id**

```
template<typename _CharT>
```

```
locale::id std::numpunct< _CharT >::id [static], [inherited]
```

Numpunct facet id.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

**5.733 \_\_gnu\_parallel::omp\_loop\_static\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for \_\_gnu\_parallel::omp\_loop\_static\_tag:



## Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

### 5.733.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

### 5.733.2 Member Function Documentation

#### [\\_\\_get\\_num\\_threads\(\)](#)

[\\_ThreadIndex \\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#) [inline], [inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

#### [set\\_num\\_threads\(\)](#)

void [\\_\\_gnu\\_parallel::parallel\\_tag::set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#) [inline], [inherited]

Set the desired number of threads.

#### Parameters

|                               |                            |
|-------------------------------|----------------------------|
| <a href="#">__num_threads</a> | Desired number of threads. |
|-------------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.734 \_\_gnu\_parallel::omp\_loop\_tag Struct Reference

#include <tags.h>

Inheritance diagram for [\\_\\_gnu\\_parallel::omp\\_loop\\_tag](#):



**Public Member Functions**

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

**5.734.1 Detailed Description**

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

**5.734.2 Member Function Documentation****\_\_get\_num\_threads()**

[\\_ThreadIndex \\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#) [inline], [inherited]

Find out desired number of threads.

**Returns**

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

**set\_num\_threads()**

void [\\_\\_gnu\\_parallel::parallel\\_tag::set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#) [inline], [inherited]

Set the desired number of threads.

**Parameters**

|                               |                            |
|-------------------------------|----------------------------|
| <a href="#">__num_threads</a> | Desired number of threads. |
|-------------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

**5.735 std::once\_flag Struct Reference**

```
#include <mutex>
```

**Public Member Functions**

- [once\\_flag](#) (const [once\\_flag](#) &)=delete
- [once\\_flag](#) & [operator=](#) (const [once\\_flag](#) &)=delete

**Friends**

- [template<typename \\_Callable, typename... \\_Args>](#)  
void [call\\_once](#) ([once\\_flag](#) &\_\_once, [\\_Callable](#) &&\_\_f, [\\_Args](#) &&... \_\_args)

**5.735.1 Detailed Description**

Flag type used by `std::call_once`.

### 5.735.2 Constructor & Destructor Documentation

#### once\_flag()

```
std::once_flag::once_flag (
    const once_flag & ) [delete]
```

Deleted copy constructor.

### 5.735.3 Member Function Documentation

#### operator=()

```
once_flag & std::once_flag::operator= (
    const once_flag & ) [delete]
```

Deleted assignment operator.

### 5.735.4 Friends And Related Symbol Documentation

#### call\_once

```
template<typename _Callable, typename... _Args>
void call_once (
    once_flag & __once,
    _Callable && __f,
    _Args &&... __args) [friend]
```

Invoke a callable and synchronize with other calls using the same flag.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 5.736 std::experimental::fundamentals\_v1::optional<\_Tp> Class Template Reference

```
#include <optional>
```

### Public Types

- using [value\\_type](#)

### Public Member Functions

- template<typename \_Up = \_Tp, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< [is\\_same](#)< [optional](#)< \_Tp >, [decay\\_t](#)< \_Up > >, [is\\_constructible](#)< \_Tp, \_Up && >, [is\\_convertible](#)< \_Up &&, \_Tp > >::value, bool > = true> [constexpr optional](#) (\_Up &&\_\_t)
- template<typename \_Up = \_Tp, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< [is\\_same](#)< [optional](#)< \_Tp >, [decay\\_t](#)< \_Up > >, [is\\_constructible](#)< \_Tp, \_Up && >, \_\_not\_< [is\\_convertible](#)< \_Up &&, \_Tp > >::value, bool > = false> [constexpr optional](#) (\_Up &&\_\_t)
- template<typename \_Up, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< [is\\_same](#)< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, const \_Up & >, [is\\_convertible](#)< const \_Up &, \_Tp >, \_\_not\_< [converts\\_from\\_optional](#)< \_Tp, \_Up > >::value, bool > = true> [constexpr optional](#) (const [optional](#)< \_Up > &\_\_t)
- template<typename \_Up, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< [is\\_same](#)< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, const \_Up & >, \_\_not\_< [is\\_convertible](#)< const \_Up &, \_Tp > >, \_\_not\_< [converts\\_from\\_optional](#)< \_Tp, \_Up > >::value, bool > = false> [constexpr optional](#) (const [optional](#)< \_Up > &\_\_t)
- template<typename \_Up, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< [is\\_same](#)< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, \_Up && >, [is\\_convertible](#)< \_Up &&, \_Tp >, \_\_not\_< [converts\\_from\\_optional](#)< \_Tp, \_Up > >::value, bool > = true> [constexpr optional](#) ([optional](#)< \_Up > &&\_\_t)

- template<typename \_Up, [enable\\_if\\_t](#)< \_\_and< \_\_not< [is\\_same](#)< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, \_Up && >, \_\_not< [is\\_convertible](#)< \_Up &&, \_Tp > >, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up > > >::value, bool > = false> constexpr **optional** ([optional](#)< \_Up > &&\_\_t)
- template<typename... \_Args>  
[enable\\_if\\_t](#)< [is\\_constructible](#)< \_Tp, \_Args &&... >::value > **emplace** (\_Args &&... \_\_args)
- template<typename \_Up, typename... \_Args>  
[enable\\_if\\_t](#)< [is\\_constructible](#)< \_Tp, [initializer\\_list](#)< \_Up > &, \_Args &&... >::value > **emplace** ([initializer\\_list](#)< \_Up > \_\_il, \_Args &&... \_\_args)
- constexpr **operator bool** () const noexcept
- constexpr \_Tp & **operator\*** () &
- constexpr \_Tp && **operator\*** () &&
- constexpr const \_Tp & **operator\*** () const &
- constexpr const \_Tp && **operator\*** () const &&
- \_Tp \* **operator->** ()
- constexpr const \_Tp \* **operator->** () const
- template<typename \_Up = \_Tp>  
[enable\\_if\\_t](#)< \_\_and< \_\_not< [is\\_same](#)< [optional](#)< \_Tp >, [decay\\_t](#)< \_Up > > >, [is\\_constructible](#)< \_Tp, \_Up >, \_\_not< \_\_and< [is\\_scalar](#)< \_Tp >, [is\\_same](#)< \_Tp, [decay\\_t](#)< \_Up > > > >, [is\\_assignable](#)< \_Tp &, \_Up > >::value, [optional](#) & > **operator=** (\_Up &&\_\_u)
- template<typename \_Up>  
[enable\\_if\\_t](#)< \_\_and< \_\_not< [is\\_same](#)< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, const \_Up & >, [is\\_assignable](#)< \_Tp &, \_Up >, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up > >, \_\_not< \_\_assigns\_from\_optional< \_Tp, \_Up > > >::value, [optional](#) & > **operator=** (const [optional](#)< \_Up > &\_\_u)
- [optional](#) & **operator=** ([nullopt\\_t](#)) noexcept
- template<typename \_Up>  
[enable\\_if\\_t](#)< \_\_and< \_\_not< [is\\_same](#)< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, \_Up >, [is\\_assignable](#)< \_Tp &, \_Up >, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up > >, \_\_not< \_\_assigns\_from\_optional< \_Tp, \_Up > > >::value, [optional](#) & > **operator=** ([optional](#)< \_Up > &&\_\_u)
- void **swap** ([optional](#) &\_\_other) noexcept([is\\_nothrow\\_move\\_constructible](#)< \_Tp >()) &&\_\_is\_nothrow\_swappable< \_Tp >::value)
- constexpr \_Tp & **value** () &
- constexpr \_Tp && **value** () &&
- constexpr const \_Tp & **value** () const &
- constexpr const \_Tp && **value** () const &&
- template<typename \_Up>  
\_Tp **value\_or** (\_Up &&\_\_u) &&
- template<typename \_Up>  
constexpr \_Tp **value\_or** (\_Up &&\_\_u) const &

### 5.736.1 Detailed Description

template<typename \_Tp>

class std::experimental::fundamentals\_v1::optional<\_Tp>

Class template for optional values.

The documentation for this class was generated from the following file:

- [experimental/optional](#)



### 5.737 std::ostream\_iterator< \_Tp, \_CharT, \_Traits > Class Template Reference

```
#include <stream_iterator.h>
```

Inheritance diagram for std::ostream\_iterator< \_Tp, \_CharT, \_Traits >:



#### Public Types

- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)
- using [difference\\_type](#)
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
- typedef [basic\\_ostream< \\_CharT, \\_Traits >](#) [ostream\\_type](#)

#### Public Member Functions

- [ostream\\_iterator](#) (const [ostream\\_iterator](#) &\_\_obj) noexcept
- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s) noexcept
- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s, const [\\_CharT](#) \*\_\_c) noexcept
- [ostream\\_iterator](#) & [operator\\*](#) () noexcept
- [ostream\\_iterator](#) & [operator++](#) () noexcept
- [ostream\\_iterator](#) & [operator++](#) (int) noexcept
- [ostream\\_iterator](#) & [operator=](#) (const [\\_Tp](#) &\_\_value)
- [ostream\\_iterator](#) & [operator=](#) (const [ostream\\_iterator](#) &)=default

#### 5.737.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::ostream_iterator< _Tp, _CharT, _Traits >
```

Provides output iterator semantics for streams.

This class provides an iterator to write to an ostream. The type `Tp` is the only type written by this iterator and there must be an `operator<<(Tp)` defined.

## Template Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>_Tp</code>     | The type to write to the ostream. |
| <code>_CharT</code>  | The ostream char_type.            |
| <code>_Traits</code> | The ostream char_traits.          |

## 5.737.2 Member Typedef Documentation

**char\_type**

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
```

```
typedef _CharT std::ostream_iterator< _Tp, _CharT, _Traits >::char_type
```

Public typedef.

**difference\_type**

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
```

```
using std::ostream_iterator< _Tp, _CharT, _Traits >::difference_type
```

Public typedef.

**iterator\_category**

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_↵  
_category [inherited]
```

One of the [tag types](#).

**ostream\_type**

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
```

```
typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_↵  
type
```

Public typedef.

**pointer**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

**reference**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value\_type.

**traits\_type**

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
```

```
typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type
```

Public typedef.

**value\_type**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
```

The type "pointed to" by the iterator.

### 5.737.3 Constructor & Destructor Documentation

#### ostream\_iterator() [1/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
    ostream_type & __s) [inline], [noexcept]
```

Construct from an ostream.

References [std::\\_\\_addressof\(\)](#).

Referenced by [ostream\\_iterator\(\)](#).

#### ostream\_iterator() [2/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
    ostream_type & __s,
    const _CharT * __c) [inline], [noexcept]
```

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

#### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <a href="#">_s</a> | Underlying ostream to write to.   |
| <a href="#">_c</a> | CharT delimiter string to insert. |

References [std::\\_\\_addressof\(\)](#).

#### ostream\_iterator() [3/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
    const ostream_iterator< _Tp, _CharT, _Traits > & __obj) [inline], [noexcept]
```

Copy constructor.

References [ostream\\_iterator\(\)](#).

### 5.737.4 Member Function Documentation

#### operator=()

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
ostream_iterator & std::ostream_iterator< _Tp, _CharT, _Traits >::operator= (
    const _Tp & __value) [inline]
```

Writes *value* to underlying ostream using `operator<<`. If constructed with delimiter string, writes delimiter to ostream. The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 5.738 std::experimental::fundamentals\_v2::ostream\_joiner<\_DelimT, \_CharT, \_Traits > Class Template Reference

```
#include <iterator>
```

**Public Types**

- typedef `_CharT` **char\_type**
- typedef void **difference\_type**
- typedef [output\\_iterator\\_tag](#) **iterator\_category**
- typedef [basic\\_ostream](#)< `_CharT`, `_Traits` > **ostream\_type**
- typedef void **pointer**
- typedef void **reference**
- typedef `_Traits` **traits\_type**
- typedef void **value\_type**

**Public Member Functions**

- **ostream\_joiner** ([ostream\\_type](#) &\_\_os, `_DelimT` &&\_\_delimiter) noexcept(is\_nothrow\_move\_constructible\_v< `_DelimT` >)
- **ostream\_joiner** ([ostream\\_type](#) &\_\_os, const `_DelimT` &\_\_delimiter) noexcept(is\_nothrow\_copy\_constructible\_v< `_DelimT` >)
- [ostream\\_joiner](#) & **operator\*** () noexcept
- [ostream\\_joiner](#) & **operator++** () noexcept
- [ostream\\_joiner](#) & **operator++** (int) noexcept
- template<typename `_Tp`>  
[ostream\\_joiner](#) & **operator=** (const `_Tp` &\_\_value)

**5.738.1 Detailed Description**

template<typename `_DelimT`, typename `_CharT` = char, typename `_Traits` = char\_traits<`_CharT`>>  
class std::experimental::fundamentals\_v2::ostream\_joiner< `_DelimT`, `_CharT`, `_Traits` >

Output iterator that inserts a delimiter between elements.

The documentation for this class was generated from the following file:

- [experimental/iterator](#)

**5.739 std::ostreambuf\_iterator< \_CharT, \_Traits > Class Template Reference**

```
#include <ostreambuf_iterator.h>
```

Inheritance diagram for std::ostreambuf\_iterator< `_CharT`, `_Traits` >:



## Public Types

- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)
  
- using [difference\\_type](#)
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
- typedef [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > [streambuf\\_type](#)
- typedef [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > [ostream\\_type](#)

## Public Member Functions

- [ostreambuf\\_iterator](#) ([ostream\\_type](#) &\_\_s) noexcept
- [ostreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) noexcept
- [ostreambuf\\_iterator](#) & [M\\_put](#) (const [\\_CharT](#) \*\_\_ws, [streamsize](#) \_\_len)
- bool [failed](#) () const noexcept
- [ostreambuf\\_iterator](#) & [operator\\*](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) (int)
- [ostreambuf\\_iterator](#) & [operator=](#) ([\\_CharT](#) \_\_c)

## Friends

- template<typename [\\_CharT2](#)>  
    \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< [\\_CharT2](#) >::\_\_value, [ostreambuf\\_iterator](#)< [\\_CharT2](#) >::\_\_type >::\_\_type copy  
    ([istreambuf\\_iterator](#)< [\\_CharT2](#) >, [istreambuf\\_iterator](#)< [\\_CharT2](#) >, [ostreambuf\\_iterator](#)< [\\_CharT2](#) >)

### 5.739.1 Detailed Description

```
template<typename \_CharT, typename \_Traits>
class std::ostreambuf_iterator< \_CharT, \_Traits >
```

Provides output iterator semantics for streambufs.

### 5.739.2 Member Typedef Documentation

#### [char\\_type](#)

```
template<typename \_CharT, typename \_Traits>
typedef \_CharT std::ostreambuf\_iterator< \_CharT, \_Traits >::char_type
Public typedefs.
```

#### [difference\\_type](#)

```
template<typename \_CharT, typename \_Traits>
using std::ostreambuf\_iterator< \_CharT, \_Traits >::difference_type
Public typedefs.
```

#### [iterator\\_category](#)

```
typedef output\_iterator\_tag std::iterator< output\_iterator\_tag, void, void, void, void >::iterator←
_category [inherited]
One of the tag types.
```

**ostream\_type**

```
template<typename _CharT, typename _Traits>
typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::ostream_type
Public typedefs.
```

**pointer**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
This type represents a pointer-to-value_type.
```

**reference**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
This type represents a reference-to-value_type.
```

**streambuf\_type**

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::streambuf←
_type
Public typedefs.
```

**traits\_type**

```
template<typename _CharT, typename _Traits>
typedef _Traits std::ostreambuf_iterator< _CharT, _Traits >::traits_type
Public typedefs.
```

**value\_type**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
The type "pointed to" by the iterator.
```

**5.739.3 Constructor & Destructor Documentation****ostreambuf\_iterator() [1/2]**

```
template<typename _CharT, typename _Traits>
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
    ostream_type & __s) [inline], [noexcept]
```

Construct output iterator from ostream.

**ostreambuf\_iterator() [2/2]**

```
template<typename _CharT, typename _Traits>
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
    streambuf_type * __s) [inline], [noexcept]
```

Construct output iterator from streambuf.

**5.739.4 Member Function Documentation****failed()**

```
template<typename _CharT, typename _Traits>
bool std::ostreambuf_iterator< _CharT, _Traits >::failed () const [inline], [nodiscard], [noexcept]
Return true if previous operator=( ) failed.
```

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::operator<<\(\)](#).

### **operator\*()**

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator* () [inline], [nodiscard]
Return *this.
```

### **operator++() [1/2]**

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator++ () [inline]
Return *this.
```

### **operator++() [2/2]**

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator++ (
    int ) [inline]
Return *this.
```

### **operator=()**

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator= (
    _CharT __c) [inline]
```

Write character to streambuf. Calls streambuf.sputc().

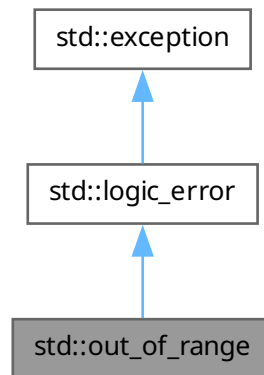
The documentation for this class was generated from the following files:

- [iosfwd](#)
- [streambuf\\_iterator.h](#)

## **5.740 std::out\_of\_range Class Reference**

```
#include <stdexcept>
```

Inheritance diagram for `std::out_of_range`:



### Public Member Functions

- `out_of_range` (const char \*)
- `out_of_range` (const [out\\_of\\_range](#) &)=default
- `out_of_range` (const [string](#) &\_\_arg)
- `out_of_range` ([out\\_of\\_range](#) &&)=default
- `out_of_range` & `operator=` (const [out\\_of\\_range](#) &)=default
- `out_of_range` & `operator=` ([out\\_of\\_range](#) &&)=default
- virtual const char \* `what` () const noexcept

#### 5.740.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in `basic_string`).

#### 5.740.2 Member Function Documentation

##### `what()`

virtual const char \* `std::logic_error::what` () const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 5.741 `std::output_iterator_tag` Struct Reference

```
#include <stl_iterator_base_types.h>
```



### 5.741.1 Detailed Description

Marking output iterators.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 5.742 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

```
#include <ov_tree_map_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`:



### Classes

- class [cond\\_dtor](#)

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `point_const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**

- typedef `ov_tree_tag` `container_category`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `point_iterator` `iterator`
- typedef `traits_base::key_const_pointer` `key_const_pointer`
- typedef `traits_base::key_const_reference` `key_const_reference`
- typedef `traits_base::key_pointer` `key_pointer`
- typedef `traits_base::key_reference` `key_reference`
- typedef `traits_base::key_type` `key_type`
- typedef `traits_base::mapped_const_pointer` `mapped_const_pointer`
- typedef `traits_base::mapped_const_reference` `mapped_const_reference`
- typedef `traits_base::mapped_pointer` `mapped_pointer`
- typedef `traits_base::mapped_reference` `mapped_reference`
- typedef `traits_base::mapped_type` `mapped_type`
- typedef `__nothrowcopy::indicator` `no_throw_indicator`
- typedef `traits_type::node_const_iterator` `node_const_iterator`
- typedef `traits_type::node_iterator` `node_iterator`
- typedef `traits_type::node_update` `node_update`
- typedef `const_pointer` `point_const_iterator`
- typedef `pointer` `point_iterator`
- typedef `traits_base::pointer` `pointer`
- typedef `traits_base::reference` `reference`
- typedef `_Alloc::size_type` `size_type`
- typedef `integral_constant< int, Store_Hash >` `store_extra`
- typedef `stored_data< value_type, size_type, Store_Hash >` `stored_data_type`
- typedef `traits_base::value_type` `value_type`

### Public Member Functions

- `ov_tree_map` (`const Cmp_Fn &`)
- `ov_tree_map` (`const Cmp_Fn &, const node_update &`)
- `ov_tree_map` (`const ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- `iterator begin` ()
- `const_iterator begin` () `const`
- `void clear` ()
- `template<typename It>`  
`void copy_from_range` (`It, It`)
- `bool empty` () `const`
- `iterator end` ()
- `const_iterator end` () `const`
- `iterator erase` (`iterator it`)
- `bool erase` (`key_const_reference`)
- `template<typename Pred>`  
`size_type erase_if` (`Pred`)
- `point_iterator find` (`key_const_reference r_key`)
- `point_const_iterator find` (`key_const_reference r_key`) `const`
- `Cmp_Fn & get_cmp_fn` ()
- `const Cmp_Fn & get_cmp_fn` () `const`
- `std::pair< point_iterator, bool > insert` (`const_reference r_value`)
- `void join` (`ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- `point_iterator lower_bound` (`key_const_reference r_key`)
- `point_const_iterator lower_bound` (`key_const_reference r_key`) `const`

- size\_type **max\_size** () const
- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_begin** () const
- node\_iterator **node\_end** ()
- node\_const\_iterator **node\_end** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **split** (key\_const\_reference, [ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** ([ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference r\_key)
- point\_const\_iterator **upper\_bound** (key\_const\_reference r\_key) const

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**
- store\_extra **m\_store\_extra\_indicator**

#### 5.742.1 Detailed Description

**template**<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename [\\_↵](#) Alloc>

**class** [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >

Ordered-vector tree associative-container.

#### 5.742.2 Member Function Documentation

##### **node\_begin()** [1/2]

**template**<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename [\\_↵](#) Alloc>

[ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_iterator [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_begin () [inline]

Returns a node\_iterator corresponding to the node at the root of the tree.

##### **node\_begin()** [2/2]

**template**<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename [\\_↵](#) Alloc>

[ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_const\_iterator [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_begin () const [inline]

Returns a const node\_iterator corresponding to the node at the root of the tree.

##### **node\_end()** [1/2]

**template**<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename [\\_↵](#) Alloc>

[ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_iterator [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_end () [inline]

Returns a node\_iterator corresponding to a node just after a leaf of the tree.

## node\_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
```

```
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_r
```

```
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () const [inline]
```

Returns a const node\_iterator corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_.hpp](#)

## 5.743 `__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

```
#include <node_iterators.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >`:



## Public Types

- typedef [rebind\\_traits](#)< \_Alloc, typename remove\_const< Value\_Type >::type >::const\_pointer **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::const\_reference **metadata\_const\_reference**
- typedef Metadata\_Type **metadata\_type**
- typedef [rebind\\_traits](#)< \_Alloc, typename remove\_const< Value\_Type >::type >::const\_pointer **reference**
- typedef [rebind\\_traits](#)< \_Alloc, Value\_Type >::const\_pointer **value\_type**

## Public Member Functions

- **ov\_tree\_node\_const\_it\_** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_metadata\_pointer p\_metadata=0)
- [this\\_type get\\_l\\_child](#) () const

- metadata\_const\_reference **get\_metadata** () const
- [this\\_type](#) **get\_r\_child** () const
- bool **operator!=** (const [this\\_type](#) &other) const
- const\_reference **operator\*** () const
- bool **operator==** (const [this\\_type](#) &other) const

#### Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**

#### Protected Types

- typedef [rebind\\_traits](#)< \_Alloc, Metadata\_Type >::const\_pointer **const\_metadata\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Value\_Type >::const\_pointer **const\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Value\_Type >::pointer **pointer**
- typedef [ov\\_tree\\_node\\_const\\_it](#)< Value\_Type, Metadata\_Type, \_Alloc > **this\_type**

#### Static Protected Member Functions

- template<typename Ptr>  
static Ptr **mid\_pointer** (Ptr p\_begin, Ptr p\_end)

#### 5.743.1 Detailed Description

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>  
class __gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >
```

Const node reference.

#### 5.743.2 Member Function Documentation

##### **get\_l\_child()**

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>  
this\_type __gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >::get_l_↵  
child () const [inline]
```

Returns the node iterator associated with the left node.

##### **get\_r\_child()**

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>  
this\_type __gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >::get_↵  
r_child () const [inline]
```

Returns the node iterator associated with the right node.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

## 5.744 `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

`#include <node_iterators.hpp>`

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >`:



### Public Types

- typedef [rebind\\_traits](#)< \_Alloc, typename remove\_const< Value\_Type >::type >::pointer **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::const\_reference **metadata\_const\_reference**
- typedef Metadata\_Type **metadata\_type**
- typedef [rebind\\_traits](#)< \_Alloc, typename remove\_const< Value\_Type >::type >::pointer **reference**
- typedef [rebind\\_traits](#)< \_Alloc, Value\_Type >::pointer **value\_type**

### Public Member Functions

- **ov\_tree\_node\_it\_** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_↵ metadata\_pointer p\_metadata=0)
- [ov\\_tree\\_node\\_it\\_get\\_l\\_child](#) () const
- metadata\_const\_reference **get\_metadata** () const
- [ov\\_tree\\_node\\_it\\_get\\_r\\_child](#) () const
- bool **operator!=** (const [this\\_type](#) &other) const
- reference [operator\\*](#) () const
- bool **operator==** (const [this\\_type](#) &other) const

### Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**

## Static Protected Member Functions

- `template<typename Ptr>`  
`static Ptr mid_pointer (Ptr p_begin, Ptr p_end)`

### 5.744.1 Detailed Description

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >
```

Node reference.

### 5.744.2 Member Function Documentation

#### `get_l_child()`

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↵
_l_child () const [inline]
```

Returns the node reference associated with the left node.

#### `get_r_child()`

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↵
_r_child () const [inline]
```

Returns the node reference associated with the right node.

#### `operator*()`

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
reference __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* ()
const [inline]
```

Access.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

## 5.745 \_\_gnu\_pbds::ov\_tree\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::ov_tree_tag`:



#### 5.745.1 Detailed Description

Ordered-vector tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.746 `std::overflow_error` Class Reference

```
#include <stdexcept>
```



Inheritance diagram for `std::overflow_error`:



### Public Member Functions

- **overflow\_error** (const char \*)
- **overflow\_error** (const [overflow\\_error](#) &)=default
- **overflow\_error** (const [string](#) & \_\_arg)
- **overflow\_error** ([overflow\\_error](#) &&)=default
- [overflow\\_error](#) & **operator=** (const [overflow\\_error](#) &)=default
- [overflow\\_error](#) & **operator=** ([overflow\\_error](#) &&)=default
- virtual const char \* **what** () const noexcept

#### 5.746.1 Detailed Description

Thrown to indicate arithmetic overflow.

#### 5.746.2 Member Function Documentation

##### what()

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), and [std::filesystem::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 5.747 std::owner\_less< \_Tp > Struct Template Reference

#### 5.747.1 Detailed Description

```
template<typename _Tp = void>
```

```
struct std::owner_less< _Tp >
```

Primary template `owner_less`.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 5.748 `std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >` Struct Template Reference

```
#include <shared_ptr.h>
```

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool` **operator()** (`const _Tp &__lhs, const _Tp &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp &__lhs, const _Tp1 &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp1 &__lhs, const _Tp &__rhs`) `const noexcept`

#### 5.748.1 Detailed Description

```
template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >
```

Partial specialization of `owner_less` for `shared_ptr`.

#### 5.748.2 Member Typedef Documentation

##### `first_argument_type`

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### `result_type`

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type
```

##### `second_argument_type`

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

## 5.749 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference

```
#include <shared_ptr.h>
```

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- **bool operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- **bool operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- **bool operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

### 5.749.1 Detailed Description

```
template<typename _Tp>
struct std::owner_less< shared_ptr< _Tp > >
```

Partial specialization of owner\_less for shared\_ptr.

### 5.749.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:
```

- [bits/shared\\_ptr.h](#)

## 5.750 std::owner\_less< void > Struct Reference

```
#include <shared_ptr.h>
```

## Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

## Public Member Functions

- **bool operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- **bool operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- **bool operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

### 5.750.1 Detailed Description

Void specialization of owner\_less compares either shared\_ptr or weak\_ptr.

### 5.750.2 Member Typedef Documentation

#### first\_argument\_type

typedef \_Tp [std::binary\\_function](#)< \_Tp, \_Tp, bool >::first\_argument\_type [inherited]  
 first\_argument\_type is the type of the first argument

#### result\_type

typedef bool [std::binary\\_function](#)< \_Tp, \_Tp, bool >::result\_type [inherited]  
 result\_type is the return type

#### second\_argument\_type

typedef \_Tp [std::binary\\_function](#)< \_Tp, \_Tp, bool >::second\_argument\_type [inherited]  
 second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 5.751 std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference

```
#include <shared_ptr.h>
```

### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

### 5.751.1 Detailed Description

**template**<typename \_Tp>

**struct** [std::experimental::fundamentals\\_v2::owner\\_less](#)< [weak\\_ptr](#)< \_Tp > >

Partial specialization of [owner\\_less](#) for [weak\\_ptr](#).

### 5.751.2 Member Typedef Documentation

#### first\_argument\_type

typedef \_Tp [std::binary\\_function](#)< \_Tp, \_Tp, bool >::first\_argument\_type [inherited]  
 first\_argument\_type is the type of the first argument

#### result\_type

typedef bool [std::binary\\_function](#)< \_Tp, \_Tp, bool >::result\_type [inherited]  
 result\_type is the return type

**second\_argument\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

**5.752 std::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference**

```
#include <shared_ptr.h>
```

**Public Types**

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

**Public Member Functions**

- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

**5.752.1 Detailed Description**

```
template<typename _Tp>
```

```
struct std::owner_less< weak_ptr< _Tp > >
```

Partial specialization of owner\_less for weak\_ptr.

**5.752.2 Member Typedef Documentation****first\_argument\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

**result\_type**

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
```

result\_type is the return type

**second\_argument\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

**5.753 std::packaged\_task< \_Res(\_ArgTypes...)> Class Template Reference**

```
#include <future>
```

**Public Member Functions**

- `template<typename _Fn, typename = __not_same<_Fn>>`  
`packaged_task` (`_Fn &&_fn`)
- `packaged_task` (`const packaged_task &`)=delete
- `packaged_task` (`packaged_task &&__other`) noexcept
- `future<_Res>` `get_future` ()
- void `make_ready_at_thread_exit` (`_ArgTypes... __args`)
- void `operator()` (`_ArgTypes... __args`)
- `packaged_task & operator=` (`const packaged_task &`)=delete
- `packaged_task & operator=` (`packaged_task &&__other`) noexcept
- void `reset` ()
- void `swap` (`packaged_task &__other`) noexcept
- bool `valid` () const noexcept

**5.753.1 Detailed Description**

`template<typename _Res, typename... _ArgTypes>`  
`class std::packaged_task<_Res(_ArgTypes...)>`

`packaged_task`

The documentation for this class was generated from the following file:

- `future`

**5.754 `__gnu_cxx::pair<_T1, _T2>` Struct Template Reference**

```
#include <utility>
```

**Public Types**

- typedef `_T1` `first_type`
- typedef `_T2` `second_type`

**Public Member Functions**

- `template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if<__and<__is_implicitly_default_constructible<_U1>, __is_implicitly_default_constructible<_U2>>::value, bool>::type = true>`  
`constexpr pair` ()
- `template<typename _U2, __enable_if_t<__and<is_pointer<_T1>, __not<is_reference<_U2>>, is_constructible<_T2, _U2>, __not<is_constructible<_T2, const _U2 &>>, is_convertible<_U2, _T2>>::value, bool> = true>`  
`constexpr pair` (`__zero_as_null_pointer_constant, _U2 &&__y,...`)
- `template<typename _U2, __enable_if_t<__and<is_pointer<_T1>, __not<is_reference<_U2>>, is_constructible<_T2, _U2>, __not<is_constructible<_T2, const _U2 &>>, __not<is_convertible<_U2, _T2>>::value, bool> = false>`  
`constexpr pair` (`__zero_as_null_pointer_constant, _U2 &&__y,...`)
- `template<typename _U1, __enable_if_t<__and<__not<is_reference<_U1>>, is_pointer<_T2>, is_constructible<_T1, _U1>, __not<is_constructible<_T1, const _U1 &>>, is_convertible<_U1, _T1>>::value, bool> = true>`  
`constexpr pair` (`_U1 &&__x, __zero_as_null_pointer_constant,...`)
- `template<typename _U1, __enable_if_t<__and<__not<is_reference<_U1>>, is_pointer<_T2>, is_constructible<_T1, _U1>, __not<is_constructible<_T1, const _U1 &>>, __not<is_convertible<_U1, _T1>>::value, bool> = false>`  
`constexpr pair` (`_U1 &&__x, __zero_as_null_pointer_constant,...`)
- `template<typename _U1, typename _U2, typename enable_if<__PCCP::template _MoveConstructiblePair<_U1, _U2>() &&__PCCP::template _ImplicitlyMoveConvertiblePair<_U1, _U2>(), bool>::type = true>`  
`constexpr pair` (`_U1 &&__x, _U2 &&__y`)

- `template<typename _U1, typename _U2, typename enable_if< _PCCP::template _MoveConstructiblePair< _U1, _U2 >() &&!_PCCP::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = false>`  
`constexpr pair ( _U1 &&__x, _U2 &&__y)`
- `template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = true>`  
`constexpr pair (const _T1 &__a, const _T2 &__b)`
- `template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&!_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = false>`  
`constexpr pair (const _T1 &__a, const _T2 &__b)`
- `constexpr pair (const pair &)=default`
- `template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _ConstructiblePair< _U1, _U2 >() &&_PCCFP< _U1, _U2 >::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = true>`  
`constexpr pair (const pair< _U1, _U2 > &__p)`
- `template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _ConstructiblePair< _U1, _U2 >() &&!_PCCFP< _U1, _U2 >::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = false>`  
`constexpr pair (const pair< _U1, _U2 > &__p)`
- `constexpr pair (pair &)=default`
- `template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _MoveConstructiblePair< _U1, _U2 >() &&_PCCFP< _U1, _U2 >::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = true>`  
`constexpr pair (pair< _U1, _U2 > &&__p)`
- `template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _MoveConstructiblePair< _U1, _U2 >() &&!_PCCFP< _U1, _U2 >::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = false>`  
`constexpr pair (pair< _U1, _U2 > &&__p)`
- `template<typename... _Args1, typename... _Args2>`  
`constexpr pair (piecewise_construct_t, tuple< _Args1... >, tuple< _Args2... >)`
- `pair & operator= ( __conditional_t< __and< is_copy_assignable< _T1 >, is_copy_assignable< _T2 > >::value, const pair &, const __nonesuch & > __p)`
- `pair & operator= ( __conditional_t< __and< is_move_assignable< _T1 >, is_move_assignable< _T2 > >::value, pair &&, __nonesuch && > __p) noexcept(__and< is_nothrow_move_assignable< _T1 >, is_nothrow_move_assignable< _T2 > >::value)`
- `template<typename _U1, typename _U2>`  
`enable_if< __and< is_assignable< _T1 &, const _U1 & >, is_assignable< _T2 &, const _U2 & > >::value, pair & >::type operator= (const pair< _U1, _U2 > &__p)`
- `template<typename _U1, typename _U2>`  
`enable_if< __and< is_assignable< _T1 &, _U1 && >, is_assignable< _T2 &, _U2 && > >::value, pair & >::type operator= (pair< _U1, _U2 > &&__p)`
- `constexpr void swap (pair &__p) noexcept(__and< __is_nothrow_swappable< _T1 >, __is_nothrow_swappable< _T2 > >::value)`

## Public Attributes

- `_T1 first`
- `_T2 second`

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _T1, typename _T2>`  
`pair ( _T1, _T2) -> pair< _T1, _T2 >`
- `template<typename _T1, typename _T2>`  
`constexpr enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`

- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)`

### 5.754.1 Detailed Description

`template<typename _T1, typename _T2>`  
`struct __gnu_cxx::pair<_T1, _T2>`

Struct holding two objects of arbitrary type.

#### Template Parameters

|                  |                        |
|------------------|------------------------|
| <code>_T1</code> | Type of first object.  |
| <code>_T2</code> | Type of second object. |

<https://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>

### 5.754.2 Member Typedef Documentation

#### `first_type`

`template<typename _T1, typename _T2>`  
`typedef _T1 std::pair< _T1, _T2 >::first_type`

The type of the first member.

#### `second_type`

`template<typename _T1, typename _T2>`  
`typedef _T2 std::pair< _T1, _T2 >::second_type`

The type of the second member.

### 5.754.3 Constructor & Destructor Documentation

#### `pair()` [1/5]

`template<typename _T1, typename _T2>`  
`std::pair< _T1, _T2 >::pair (`  
`const pair< _T1, _T2 > & ) [constexpr], [default]`

Copy constructor.

#### `pair()` [2/5]

`template<typename _T1, typename _T2>`  
`std::pair< _T1, _T2 >::pair (`  
`pair< _T1, _T2 > && ) [constexpr], [default]`

Move constructor.

#### `pair()` [3/5]

`template<typename _T1, typename _T2>`  
`template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< __and< __is_implicitly_`  
`default_constructible< _U1 >, __is_implicitly_default_constructible< _U2 > > ::value, bool >=`  
`::type = true>`  
`std::pair< _T1, _T2 >::pair () [inline], [constexpr]`



The default constructor creates `first` and `second` using their respective default constructors.

#### **pair()** [4/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = true>
std::pair< _T1, _T2 >::pair (
    const _T1 & __a,
    const _T2 & __b) [inline], [constexpr]
```

Construct from two const lvalues, allowing implicit conversions.

#### **pair()** [5/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&!_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = false>
std::pair< _T1, _T2 >::pair (
    const _T1 & __a,
    const _T2 & __b) [inline], [explicit], [constexpr]
```

Construct from two const lvalues, disallowing implicit conversions.

### **5.754.4 Member Function Documentation**

#### **swap()**

```
template<typename _T1, typename _T2>
void std::pair< _T1, _T2 >::swap (
    pair< _T1, _T2 > & __p) [inline], [constexpr], [noexcept]
```

Swap the first members and then the second members.

### **5.754.5 Member Data Documentation**

#### **first**

```
template<typename _T1, typename _T2>
_T1 std::pair< _T1, _T2 >::first
```

The first member.

#### **second**

```
template<typename _T1, typename _T2>
_T2 std::pair< _T1, _T2 >::second
```

The second member.

The documentation for this struct was generated from the following files:

- [bits/stl\\_iterator.h](#)
- [stl\\_pair.h](#)
- [tuple](#)

### **5.755 std::pair< \_T1, \_T2 > Struct Template Reference**

```
#include <utility>
```

**Public Types**

- typedef `_T1` [first\\_type](#)
- typedef `_T2` [second\\_type](#)

**Public Member Functions**

- template<typename `_U1` = `_T1`, typename `_U2` = `_T2`, typename [enable\\_if](#)< `__and`< `__is_implicitly_default_constructible`< `_U1` >, `__is_implicitly_default_constructible`< `_U2` > >::value, bool >::type = true>  
constexpr [pair](#) ()
- template<typename `_U2`, `__enable_if_t`< `__and`< [is\\_pointer](#)< `_T1` >, `__not`< [is\\_reference](#)< `_U2` > >, [is\\_constructible](#)< `_T2`, `_U2` >, `__not`< [is\\_constructible](#)< `_T2`, const `_U2` & > >, [is\\_convertible](#)< `_U2`, `_T2` > >::value, bool > = true>  
constexpr [pair](#) (`__zero_as_null_pointer_constant`, `_U2` && `_y`,...)
- template<typename `_U2`, `__enable_if_t`< `__and`< [is\\_pointer](#)< `_T1` >, `__not`< [is\\_reference](#)< `_U2` > >, [is\\_constructible](#)< `_T2`, `_U2` >, `__not`< [is\\_constructible](#)< `_T2`, const `_U2` & > >, `__not`< [is\\_convertible](#)< `_U2`, `_T2` > > >::value, bool > = false>  
constexpr [pair](#) (`__zero_as_null_pointer_constant`, `_U2` && `_y`,...)
- template<typename `_U1`, `__enable_if_t`< `__and`< `__not`< [is\\_reference](#)< `_U1` > >, [is\\_pointer](#)< `_T2` >, [is\\_constructible](#)< `_T1`, `_U1` >, `__not`< [is\\_constructible](#)< `_T1`, const `_U1` & > >, [is\\_convertible](#)< `_U1`, `_T1` > >::value, bool > = true>  
constexpr [pair](#) (`_U1` && `_x`, `__zero_as_null_pointer_constant`,...)
- template<typename `_U1`, `__enable_if_t`< `__and`< `__not`< [is\\_reference](#)< `_U1` > >, [is\\_pointer](#)< `_T2` >, [is\\_constructible](#)< `_T1`, `_U1` >, `__not`< [is\\_constructible](#)< `_T1`, const `_U1` & > >, `__not`< [is\\_convertible](#)< `_U1`, `_T1` > > >::value, bool > = false>  
constexpr [pair](#) (`_U1` && `_x`, `__zero_as_null_pointer_constant`,...)
- template<typename `_U1`, typename `_U2`, typename [enable\\_if](#)< `_PCCP`::template `_MoveConstructiblePair`< `_U1`, `_U2` >() && `_PCCP`::template `_ImplicitlyMoveConvertiblePair`< `_U1`, `_U2` >(), bool >::type = true>  
constexpr [pair](#) (`_U1` && `_x`, `_U2` && `_y`)
- template<typename `_U1`, typename `_U2`, typename [enable\\_if](#)< `_PCCP`::template `_MoveConstructiblePair`< `_U1`, `_U2` >() && !`_PCCP`::template `_ImplicitlyMoveConvertiblePair`< `_U1`, `_U2` >(), bool >::type = false>  
constexpr [pair](#) (`_U1` && `_x`, `_U2` && `_y`)
- template<typename `_U1` = `_T1`, typename `_U2` = `_T2`, typename [enable\\_if](#)< `_PCCP`::template `_ConstructiblePair`< `_U1`, `_U2` >() && `_PCCP`::template `_ImplicitlyConvertiblePair`< `_U1`, `_U2` >(), bool >::type = true>  
constexpr [pair](#) (const `_T1` & `_a`, const `_T2` & `_b`)
- template<typename `_U1` = `_T1`, typename `_U2` = `_T2`, typename [enable\\_if](#)< `_PCCP`::template `_ConstructiblePair`< `_U1`, `_U2` >() && !`_PCCP`::template `_ImplicitlyConvertiblePair`< `_U1`, `_U2` >(), bool >::type = false>  
constexpr [pair](#) (const `_T1` & `_a`, const `_T2` & `_b`)
- constexpr [pair](#) (const [pair](#) &)=default
- template<typename `_U1`, typename `_U2`, typename [enable\\_if](#)< `_PCCFP`< `_U1`, `_U2` >::template `_ConstructiblePair`< `_U1`, `_U2` >() && `_PCCFP`< `_U1`, `_U2` >::template `_ImplicitlyConvertiblePair`< `_U1`, `_U2` >(), bool >::type = true>  
constexpr [pair](#) (const [pair](#)< `_U1`, `_U2` > & `_p`)
- template<typename `_U1`, typename `_U2`, typename [enable\\_if](#)< `_PCCFP`< `_U1`, `_U2` >::template `_ConstructiblePair`< `_U1`, `_U2` >() && !`_PCCFP`< `_U1`, `_U2` >::template `_ImplicitlyConvertiblePair`< `_U1`, `_U2` >(), bool >::type = false>  
constexpr [pair](#) (const [pair](#)< `_U1`, `_U2` > & `_p`)
- constexpr [pair](#) ([pair](#) &&)=default
- template<typename `_U1`, typename `_U2`, typename [enable\\_if](#)< `_PCCFP`< `_U1`, `_U2` >::template `_MoveConstructiblePair`< `_U1`, `_U2` >() && `_PCCFP`< `_U1`, `_U2` >::template `_ImplicitlyMoveConvertiblePair`< `_U1`, `_U2` >(), bool >::type = true>  
constexpr [pair](#) ([pair](#)< `_U1`, `_U2` > && `_p`)
- template<typename `_U1`, typename `_U2`, typename [enable\\_if](#)< `_PCCFP`< `_U1`, `_U2` >::template `_MoveConstructiblePair`< `_U1`, `_U2` >() && !`_PCCFP`< `_U1`, `_U2` >::template `_ImplicitlyMoveConvertiblePair`< `_U1`, `_U2` >(), bool >::type = false>  
constexpr [pair](#) ([pair](#)< `_U1`, `_U2` > && `_p`)
- template<typename... `_Args1`, typename... `_Args2`>  
constexpr [pair](#) ([piecewise\\_construct\\_t](#), `tuple`< `_Args1`... >, `tuple`< `_Args2`... >)
- [pair](#) & **operator=** (`__conditional_t`< `__and`< [is\\_copy\\_assignable](#)< `_T1` >, [is\\_copy\\_assignable](#)< `_T2` > > && `__value`, const [pair](#) &, const `__nonesuch` & > `_p`)

- `pair & operator=` (`__conditional_t< __and_< is_move_assignable< _T1 >, is_move_assignable< _T2 > >::value, pair &&, __nonexistent && > __p`) `noexcept(__and_< is_nothrow_move_assignable< _T1 >, is_nothrow_move_assignable< _T2 > >::value)`
- `template<typename _U1, typename _U2>`  
`enable_if< __and_< is_assignable< _T1 &, const _U1 & >, is_assignable< _T2 &, const _U2 & > >::value, pair & >::type operator=` (`const pair< _U1, _U2 > & __p`)
- `template<typename _U1, typename _U2>`  
`enable_if< __and_< is_assignable< _T1 &, _U1 && >, is_assignable< _T2 &, _U2 && > >::value, pair & >::type operator=` (`pair< _U1, _U2 > && __p`)
- `constexpr void swap` (`pair & __p`) `noexcept(__and_< __is_nothrow_swappable< _T1 >, __is_nothrow_swappable< _T2 > >::value)`

## Public Attributes

- `_T1` `first`
- `_T2` `second`

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _T1, typename _T2>`  
`pair` (`_T1, _T2`) -> `pair< _T1, _T2 >`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr bool operator==` (`const pair< _T1, _T2 > & __x, const pair< _U1, _U2 > & __y`)
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=>` (`const pair< _T1, _T2 > & __x, const pair< _U1, _U2 > & __y`)
- `template<typename _T1, typename _T2>`  
`constexpr enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap` (`pair< _T1, _T2 > & __x, pair< _T1, _T2 > & __y`) `noexcept(noexcept(__x.swap(__y)))`

### 5.755.1 Detailed Description

`template<typename _T1, typename _T2>`  
**struct** `std::pair< _T1, _T2 >`

Struct holding two objects of arbitrary type.

#### Template Parameters

|                  |                        |
|------------------|------------------------|
| <code>_T1</code> | Type of first object.  |
| <code>_T2</code> | Type of second object. |

<https://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>

### 5.755.2 Member Typedef Documentation

#### `first_type`

`template<typename _T1, typename _T2>`  
`typedef _T1` `std::pair< _T1, _T2 >::first_type`  
 The type of the `first` member.

**second\_type**

```
template<typename _T1, typename _T2>
typedef _T2 std::pair< _T1, _T2 >::second_type
```

The type of the second member.

**5.755.3 Constructor & Destructor Documentation****pair() [1/5]**

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
    const pair< _T1, _T2 > & ) [constexpr], [default]
```

Copy constructor.

**pair() [2/5]**

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
    pair< _T1, _T2 > && ) [constexpr], [default]
```

Move constructor.

**pair() [3/5]**

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< __and< __is_implicitly_↵
default_constructible< _U1 >, __is_implicitly_default_constructible< _U2 > > ::value, bool >↵
::type = true>
std::pair< _T1, _T2 >::pair () [inline], [constexpr]
```

The default constructor creates first and second using their respective default constructors.

**pair() [4/5]**

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _Constructible↵
Pair< _U1, _U2 >() &&_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type =
true>
std::pair< _T1, _T2 >::pair (
    const _T1 & __a,
    const _T2 & __b) [inline], [constexpr]
```

Construct from two const lvalues, allowing implicit conversions.

**pair() [5/5]**

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _Constructible↵
Pair< _U1, _U2 >() &&!_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type =
false>
std::pair< _T1, _T2 >::pair (
    const _T1 & __a,
    const _T2 & __b) [inline], [explicit], [constexpr]
```

Construct from two const lvalues, disallowing implicit conversions.

### 5.755.4 Member Function Documentation

#### swap()

```
template<typename _T1, typename _T2>
void std::pair< _T1, _T2 >::swap (
    pair< _T1, _T2 > & __p) [inline], [constexpr], [noexcept]
```

Swap the first members and then the second members.

Referenced by [std::pair< const \\_\\_iter\\_key\\_t< \\_InputIterator >, \\_\\_iter\\_val\\_t< \\_InputIterator > >::swap\(\)](#), [std::sub\\_match< const char \\*](#) and [std::swap\(\)](#).

### 5.755.5 Member Data Documentation

#### first

```
template<typename _T1, typename _T2>
_T1 std::pair< _T1, _T2 >::first
```

The first member.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), [std::\\_\\_sample\(\)](#), [std::set< \\_Key, \\_Cmp, polymorphic\\_allocator< \\_Key > >::insert\(\)](#), [std::pair< const \\_\\_iter\\_key\\_t< \\_InputIterator >, \\_\\_iter\\_val\\_t< \\_InputIterator > >::operator==\(\)](#), [std::pair< const \\_\\_iter\\_key\\_t< \\_InputIterator >, \\_\\_iter\\_val\\_t< \\_InputIterator > >::operator==\(\)](#), and [std::shuffle\(\)](#).

#### second

```
template<typename _T1, typename _T2>
_T2 std::pair< _T1, _T2 >::second
```

The second member.

Referenced by [std::\\_\\_sample\(\)](#), [std::set< \\_Key, \\_Cmp, polymorphic\\_allocator< \\_Key > >::insert\(\)](#), [std::pair< const \\_\\_iter\\_key\\_t< \\_InputIterator >, \\_\\_iter\\_val\\_t< \\_InputIterator > >::operator==\(\)](#), and [std::shuffle\(\)](#).

The documentation for this struct was generated from the following files:

- [bits/stl\\_iterator.h](#)
- [stl\\_pair.h](#)
- [tuple](#)

### 5.756 \_\_gnu\_pbds::detail::pairing\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

```
#include <pairing_heap.hpp>
```

Inheritance diagram for [\\_\\_gnu\\_pbds::detail::pairing\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#):



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_a::const_pointer` **const\_pointer**
- typedef `__rebind_a::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `left_child_next_sibling_heap_node_< Value_Type, null_type, _Alloc >` **node**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **pairing\_heap** (const `Cmp_Fn` &)
- **pairing\_heap** (const `pairing_heap` &)
- **iterator begin** ()
- **iterator begin** ()
- **const\_iterator begin** () const
- **const\_iterator begin** () const
- void **clear** ()
- void **clear** ()
- bool **empty** () const
- bool **empty** () const
- **iterator end** ()
- **iterator end** ()
- **const\_iterator end** () const
- **const\_iterator end** () const
- void **erase** (point\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- void **join** (`pairing_heap` &)
- size\_type **max\_size** () const
- size\_type **max\_size** () const
- void **modify** (point\_iterator, const\_reference)
- void **pop** ()
- point\_iterator **push** (const\_reference)
- size\_type **size** () const
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, `pairing_heap` &)
- void **swap** (`left_child_next_sibling_heap`< `Value_Type`, `Cmp_Fn`, `null_type`, `_Alloc` > &)
- void **swap** (`left_child_next_sibling_heap`< `Value_Type`, `Cmp_Fn`, `null_type`, `_Alloc` > &)
- void **swap** (`pairing_heap` &)
- const\_reference **top** () const

### Protected Types

- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef alloc\_traits::const\_pointer **node\_const\_pointer**
- typedef [null\\_type](#) **node\_metadata**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It>  
void **copy\_from\_range** (It, It)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- node\_pointer **prune** (Pred)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_root**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**
- size\_type **m\_size**

#### 5.756.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>  
class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >
```

Pairing heap.

The documentation for this class was generated from the following file:

- [pairing\\_heap.hpp](#)

## 5.757 \_\_gnu\_pbds::pairing\_heap\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::pairing\_heap\_tag:



### 5.757.1 Detailed Description

Pairing-heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.758 \_\_gnu\_parallel::parallel\_tag Struct Reference

```
#include <tags.h>
```



Inheritance diagram for `__gnu_parallel::parallel_tag`:



#### Public Member Functions

- [parallel\\_tag](#) ()
- [parallel\\_tag](#) ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()

- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

### 5.758.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

### 5.758.2 Constructor & Destructor Documentation

#### `parallel_tag()` [1/2]

```
__gnu_parallel::parallel_tag::parallel_tag () [inline]
```

Default constructor. Use default number of threads.

#### `parallel_tag()` [2/2]

```
__gnu_parallel::parallel_tag::parallel_tag (
    _ThreadIndex __num_threads) [inline]
```

Default constructor. Recommend number of threads to use.

#### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

### 5.758.3 Member Function Documentation

#### `__get_num_threads()`

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

#### `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
    _ThreadIndex __num_threads) [inline]
```

Set the desired number of threads.

#### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.759 `std::bernoulli_distribution::param_type` Struct Reference

```
#include <random.h>
```

## Public Types

- typedef [bernoulli\\_distribution](#) **distribution\_type**

## Public Member Functions

- **param\_type** (double \_\_p)
- double **p** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.759.1 Detailed Description

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.760 `std::binomial_distribution<_IntType>::param_type` Struct Reference

```
#include <random.h>
```

## Public Types

- typedef [binomial\\_distribution](#)<\_IntType> **distribution\_type**

## Public Member Functions

- **param\_type** (\_IntType \_\_t, double \_\_p=0.5)
- double **p** () const
- \_IntType **t** () const

## Friends

- class **binomial\_distribution**<\_IntType>
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.760.1 Detailed Description

```
template<typename _IntType = int>
struct std::binomial_distribution<_IntType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.761 `std::cauchy_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

## Public Types

- typedef [cauchy\\_distribution](#)<\_RealType> **distribution\_type**

### Public Member Functions

- `param_type` (`_RealType __a`, `_RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

### Friends

- `bool operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

#### 5.761.1 Detailed Description

```
template<typename _RealType = double>
struct std::cauchy_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.762 `std::chi_squared_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `chi_squared_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __n`)
- `_RealType n` () const

### Friends

- `bool operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

#### 5.762.1 Detailed Description

```
template<typename _RealType = double>
struct std::chi_squared_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.763 `std::discrete_distribution<_IntType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `discrete_distribution<_IntType>` `distribution_type`

### Public Member Functions

- `template<typename _InputIterator>`  
**param\_type** (`_InputIterator __wbegin, _InputIterator __wend`)
- **param\_type** (`const param_type &`)=default
- **param\_type** (`initializer_list< double > __wil`)
- `template<typename _Func>`  
**param\_type** (`size_t __nw, double __xmin, double __xmax, _Func __fw`)
- **param\_type & operator=** (`const param_type &`)=default
- `std::vector< double > probabilities` () const

### Friends

- class **discrete\_distribution**< `_IntType` >
- bool **operator==** (`const param_type &__p1, const param_type &__p2`)

#### 5.763.1 Detailed Description

```
template<typename _IntType = int>
struct std::discrete_distribution< _IntType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 5.764 std::exponential\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

#### Public Types

- typedef [exponential\\_distribution](#)< `_RealType` > **distribution\_type**

#### Public Member Functions

- **param\_type** (`_RealType __lambda`)
- `_RealType lambda` () const

#### Friends

- bool **operator==** (`const param_type &__p1, const param_type &__p2`)

#### 5.764.1 Detailed Description

```
template<typename _RealType = double>
struct std::exponential_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 5.765 std::extreme\_value\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

## Public Types

- typedef `extreme_value_distribution<_RealType>` `distribution_type`

## Public Member Functions

- `param_type` (`_RealType __a`, `_RealType __b=_RealType(1.0)`)
- `_RealType a` () const
- `_RealType b` () const

## Friends

- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.765.1 Detailed Description

```
template<typename _RealType = double>
struct std::extreme_value_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.766 `std::fisher_f_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

## Public Types

- typedef `fisher_f_distribution<_RealType>` `distribution_type`

## Public Member Functions

- `param_type` (`_RealType __m`, `_RealType __n=_RealType(1)`)
- `_RealType m` () const
- `_RealType n` () const

## Friends

- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.766.1 Detailed Description

```
template<typename _RealType = double>
struct std::fisher_f_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.767 `std::gamma_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef [gamma\\_distribution](#)< \_RealType > **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_alpha\_val, \_RealType \_\_beta\_val=\_RealType(1))
- \_RealType **alpha** () const
- \_RealType **beta** () const

### Friends

- class **gamma\_distribution**< \_RealType >
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.767.1 Detailed Description

```
template<typename _RealType = double>
struct std::gamma_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 5.768 std::geometric\_distribution< \_IntType >::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef [geometric\\_distribution](#)< \_IntType > **distribution\_type**

### Public Member Functions

- **param\_type** (double \_\_p)
- double **p** () const

### Friends

- class **geometric\_distribution**< \_IntType >
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.768.1 Detailed Description

```
template<typename _IntType = int>
struct std::geometric_distribution< _IntType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 5.769 std::lognormal\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef [lognormal\\_distribution](#)<\_RealType> **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_m, \_RealType \_\_s=\_RealType(1))
- \_RealType **m** () const
- \_RealType **s** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.769.1 Detailed Description

```
template<typename _RealType = double>
struct std::lognormal_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.770 `std::negative_binomial_distribution<_IntType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef [negative\\_binomial\\_distribution](#)<\_IntType> **distribution\_type**

### Public Member Functions

- **param\_type** (\_IntType \_\_k, double \_\_p=0.5)
- \_IntType **k** () const
- double **p** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.770.1 Detailed Description

```
template<typename _IntType = int>
struct std::negative_binomial_distribution<_IntType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.771 `std::normal_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```



### Public Types

- typedef [normal\\_distribution](#)< \_RealType > **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_mean, \_RealType \_\_stddev=\_RealType(1))
- \_RealType **mean** () const
- \_RealType **stddev** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.771.1 Detailed Description

```
template<typename _RealType = double>
struct std::normal_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 5.772 std::piecewise\_constant\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef [piecewise\\_constant\\_distribution](#)< \_RealType > **distribution\_type**

### Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW>  
**param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- **param\_type** (const [param\\_type](#) &)=default
- template<typename \_Func>  
**param\_type** ([initializer\\_list](#)< \_RealType > \_\_bi, \_Func \_\_fw)
- template<typename \_Func>  
**param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- [std::vector](#)< double > **densities** () const
- [std::vector](#)< \_RealType > **intervals** () const
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class [piecewise\\_constant\\_distribution](#)< \_RealType >

### 5.772.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_constant_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.773 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef [piecewise\\_linear\\_distribution<\\_RealType>](#) **distribution\_type**

### Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW>  
**param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- **param\_type** (const [param\\_type](#) &)=default
- template<typename \_Func>  
**param\_type** ([initializer\\_list](#)<\_RealType> \_\_bl, \_Func \_\_fw)
- template<typename \_Func>  
**param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- [std::vector](#)<double> **densities** () const
- [std::vector](#)<\_RealType> **intervals** () const
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class [piecewise\\_linear\\_distribution<\\_RealType>](#)

### 5.773.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_linear_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.774 `std::poisson_distribution<_IntType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef [poisson\\_distribution<\\_IntType>](#) **distribution\_type**

### Public Member Functions

- **param\_type** (double \_\_mean)
- double **mean** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class **poisson\_distribution**<\_IntType >

#### 5.774.1 Detailed Description

```
template<typename _IntType = int>
struct std::poisson_distribution<_IntType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 5.775 std::student\_t\_distribution<\_RealType >::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef [student\\_t\\_distribution](#)<\_RealType > **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_n)
- \_RealType **n** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.775.1 Detailed Description

```
template<typename _RealType = double>
struct std::student_t_distribution<_RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 5.776 std::uniform\_int\_distribution<\_IntType >::param\_type Struct Reference

```
#include <uniform_int_dist.h>
```

### Public Types

- typedef [uniform\\_int\\_distribution](#)<\_IntType > **distribution\_type**

### Public Member Functions

- `param_type` (`_IntType __a, _IntType __b=__gnu_cxx::__int_traits<_IntType>::__max`)
- `result_type a` () const
- `result_type b` () const

### Friends

- `bool operator!=` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- `bool operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

#### 5.776.1 Detailed Description

```
template<typename _IntType = int>
struct std::uniform_int_distribution<_IntType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [uniform\\_int\\_dist.h](#)

### 5.777 `std::uniform_real_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `uniform_real_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __a, _RealType __b=_RealType(1)`)
- `result_type a` () const
- `result_type b` () const

### Friends

- `bool operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

#### 5.777.1 Detailed Description

```
template<typename _RealType = double>
struct std::uniform_real_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 5.778 `std::weibull_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `weibull_distribution<_RealType>` `distribution_type`

## Public Member Functions

- **param\_type** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1.0))
- \_RealType **a** () const
- \_RealType **b** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.778.1 Detailed Description

**template<typename \_RealType = double>**  
**struct std::weibull\_distribution< \_RealType >::param\_type**

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 5.779 \_\_gnu\_pbds::detail::pat\_trie\_base Struct Reference

#include <pat\_trie\_base.hpp>

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base:



## Classes

- class [\\_CIter](#)
- struct [\\_Head](#)
- struct [\\_Inode](#)
- class [\\_Iter](#)
- struct [\\_Leaf](#)
- struct [\\_Metadata](#)
- struct [\\_Metadata< null\\_type, \\_Alloc >](#)
- struct [\\_Node\\_base](#)

- class [\\_Node\\_citer](#)
- class [\\_Node\\_iter](#)

## Public Types

- enum [node\\_type](#) { `i_node` , `leaf_node` , `head_node` }

### 5.779.1 Detailed Description

Base type for PATRICIA trees.

### 5.779.2 Member Enumeration Documentation

#### `node_type`

```
enum __gnu_pbds::detail::pat_trie_base::node_type
```

Three types of nodes.

`i_node` is used by `_Inode`, `leaf_node` by `_Leaf`, and `head_node` by `_Head`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.780 `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >` Class Template Reference

```
#include <pat_trie_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >`:



## Public Types

- typedef `traits_type::access_traits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `point_const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `traits_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef [pat\\_trie\\_tag](#) **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `point_iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**

- typedef traits\_base::key\_pointer **key\_pointer**
- typedef traits\_base::key\_reference **key\_reference**
- typedef traits\_base::key\_type **key\_type**
- typedef traits\_base::mapped\_const\_pointer **mapped\_const\_pointer**
- typedef traits\_base::mapped\_const\_reference **mapped\_const\_reference**
- typedef traits\_base::mapped\_pointer **mapped\_pointer**
- typedef traits\_base::mapped\_reference **mapped\_reference**
- typedef traits\_base::mapped\_type **mapped\_type**
- typedef \_\_nothrowcopy::indicator **no\_throw\_indicator**
- typedef traits\_type::node\_const\_iterator **node\_const\_iterator**
- typedef traits\_type::node\_iterator **node\_iterator**
- enum [node\\_type](#) { **i\_node** , **leaf\_node** , **head\_node** }
- typedef traits\_type::node\_update **node\_update**
- typedef traits\_type::const\_iterator **point\_const\_iterator**
- typedef traits\_type::iterator **point\_iterator**
- typedef traits\_base::pointer **pointer**
- typedef [traits\\_base::reference](#) **reference**
- typedef traits\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef [stored\\_data](#)< [value\\_type](#), size\_type, Store\_Hash > **stored\_data\_type**
- typedef [traits\\_base::value\\_type](#) **value\_type**

## Public Member Functions

- **pat\_trie\_map** (const access\_traits &)
- **pat\_trie\_map** (const [pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- const\_iterator **erase** (const\_iterator)
- const\_reverse\_iterator **erase** (const\_reverse\_iterator)
- iterator **erase** (iterator)
- bool **erase** (key\_const\_reference)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- access\_traits & **get\_access\_traits** ()
- const access\_traits & **get\_access\_traits** () const
- node\_update & **get\_node\_update** ()
- const node\_update & **get\_node\_update** () const
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference)
- void **join** ([pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const

- node\_iterator `node_begin()`
- node\_const\_iterator `node_begin()` const
- node\_iterator `node_end()`
- node\_const\_iterator `node_end()` const
- mapped\_reference `operator[]` (key\_const\_reference r\_key)
- reverse\_iterator `rbegin()`
- const\_reverse\_iterator `rbegin()` const
- reverse\_iterator `rend()`
- const\_reverse\_iterator `rend()` const
- size\_type `size()` const
- void `split` (key\_const\_reference, `pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >` &)
- void `swap` (`pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >` &)
- point\_iterator `upper_bound` (key\_const\_reference)
- point\_const\_iterator `upper_bound` (key\_const\_reference) const

### Public Attributes

- no\_throw\_indicator `m_no_throw_copies_indicator`
- no\_throw\_indicator `m_no_throw_copies_indicator`
- store\_extra `m_store_extra_indicator`
- store\_extra `m_store_extra_indicator`

### Protected Member Functions

- template<typename It>  
void `copy_from_range` (It, It)
- node\_pointer `recursive_copy_node` (node\_const\_pointer)
- void `value_swap` (`pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >` &)

#### 5.780.1 Detailed Description

`template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>`  
`class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >`

PATRICIA trie.

This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptsset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000

#### 5.780.2 Member Enumeration Documentation

##### node\_type

enum `__gnu_pbds::detail::pat_trie_base::node_type` [inherited]

Three types of nodes.

i\_node is used by `_Inode`, leaf\_node by `_Leaf`, and head\_node by `_Head`.

#### 5.780.3 Member Function Documentation

##### node\_begin() [1/2]

`template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>`  
`pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator` `__gnu_pbds::detail::pat_trie_map<`  
`Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin()` [inline]

Returns a node\_iterator corresponding to the node at the root of the tree.



**node\_begin()** [2/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin () const [inline]
```

Returns a const node\_iterator corresponding to the node at the root of the tree.

**node\_end()** [1/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () [inline]
```

Returns a node\_iterator corresponding to a node just after a leaf of the tree.

**node\_end()** [2/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () const [inline]
```

Returns a const node\_iterator corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_.hpp](#)

**5.781 \_\_gnu\_pbds::pat\_trie\_tag Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::pat\_trie\_tag:



### 5.781.1 Detailed Description

PATRICIA trie.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.782 std::experimental::filesystem::v1::path Class Reference

```
#include <fs_path.h>
```

### Classes

- class [iterator](#)

### Public Types

- typedef [iterator](#) **const\_iterator**
- typedef [std::basic\\_string](#)< value\_type > **string\_type**
- typedef char **value\_type**

## Public Member Functions

- `template<typename _InputIterator, typename _Require = __detail::_Path<_InputIterator, _InputIterator>>>`  
`path (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator, typename _Require = __detail::_Path<_InputIterator, _InputIterator>, typename _Require2 = __detail::__value_type_is_char<_InputIterator>>>`  
`path (_InputIterator __first, _InputIterator __last, const locale &__loc)`
- `template<typename _Source, typename _Require = __detail::_Path<_Source>>>`  
`path (_Source const &__source)`
- `template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _Require2 = __detail::__value_type_is_char<_Source>>>`  
`path (_Source const &__source, const locale &__loc)`
- `path (const path &__p)`
- `path (path &&__p) noexcept`
- `path (string\_type &&__source)`
- `template<typename _InputIterator>`  
`__detail::_Path<_InputIterator, _InputIterator> & append (_InputIterator __first, _InputIterator __last)`
- `template<typename _Source>`  
`__detail::_Path<_Source> & append (_Source const &__source)`
- `template<typename _InputIterator>`  
`__detail::_Path<_InputIterator, _InputIterator> & assign (_InputIterator __first, _InputIterator __last)`
- `template<typename _Source>`  
`__detail::_Path<_Source> & assign (_Source const &__source)`
- `path & assign (string\_type &&__source)`
- `iterator begin () const noexcept`
- `const value_type * c_str () const noexcept`
- `void clear () noexcept`
- `int compare (const basic\_string\_view<value_type> __s) const`
- `int compare (const path &__p) const noexcept`
- `int compare (const string\_type &__s) const`
- `int compare (const value_type * __s) const`
- `template<typename _InputIterator>`  
`__detail::_Path<_InputIterator, _InputIterator> & concat (_InputIterator __first, _InputIterator __last)`
- `template<typename _Source>`  
`__detail::_Path<_Source> & concat (_Source const &__x)`
- `bool empty () const noexcept`
- `iterator end () const noexcept`
- `path extension () const`
- `path filename () const`
- `std::string generic_string () const`
- `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>>`  
`std::basic\_string<_CharT, _Traits, _Allocator> generic_string (const _Allocator &__a=_Allocator()) const`
- `std::u16string generic_u16string () const`
- `std::u32string generic_u32string () const`
- `std::string generic_u8string () const`
- `std::wstring generic_wstring () const`
- `bool has_extension () const`
- `bool has_filename () const`
- `bool has_parent_path () const`
- `bool has_relative_path () const`
- `bool has_root_directory () const`
- `bool has_root_name () const`

- bool **has\_root\_path** () const
- bool **has\_stem** () const
- bool **is\_absolute** () const
- bool **is\_relative** () const
- **path** & **make\_preferred** ()
- const **string\_type** & **native** () const noexcept
- **operator string\_type** () const
- template<typename \_CharT>  
\_\_detail::\_Path< \_CharT \*, \_CharT \* > & **operator+=** (\_CharT \_\_x)
- template<typename \_Source>  
\_\_detail::\_Path< \_Source > & **operator+=** (\_Source const &\_\_x)
- **path** & **operator+=** (**basic\_string\_view**< value\_type > \_\_x)
- **path** & **operator+=** (const **path** &\_\_x)
- **path** & **operator+=** (const **string\_type** &\_\_x)
- **path** & **operator+=** (const value\_type \*\_\_x)
- **path** & **operator+=** (value\_type \_\_x)
- template<typename \_Source>  
\_\_detail::\_Path< \_Source > & **operator/=** (\_Source const &\_\_source)
- **path** & **operator/=** (const **path** &\_\_p)
- template<typename \_Source>  
\_\_detail::\_Path< \_Source > & **operator=** (\_Source const &\_\_source)
- **path** & **operator=** (const **path** &\_\_p)
- **path** & **operator=** (**path** &&\_\_p) noexcept
- **path** & **operator=** (**string\_type** &&\_\_source)
- **path** **parent\_path** () const
- **path** **relative\_path** () const
- **path** & **remove\_filename** ()
- **path** & **replace\_extension** (const **path** &\_\_replacement=**path**())
- **path** & **replace\_filename** (const **path** &\_\_replacement)
- **path** **root\_directory** () const
- **path** **root\_name** () const
- **path** **root\_path** () const
- **path** **stem** () const
- **std::string** **string** () const
- template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>  
**std::basic\_string**< \_CharT, \_Traits, \_Allocator > **string** (const \_Allocator &\_\_a=\_Allocator()) const
- void **swap** (**path** &\_\_rhs) noexcept
- **std::u16string** **u16string** () const
- **std::u32string** **u32string** () const
- **std::string** **u8string** () const
- **std::wstring** **wstring** () const

## Static Public Attributes

- static constexpr value\_type **preferred\_separator**

### 5.782.1 Detailed Description

A filesystem path.

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)

## 5.783 std::filesystem::path Class Reference

```
#include <filesystem>
```

### Classes

- class [iterator](#)

### Public Types

- using **const\_iterator**
- enum **format**: unsigned char { **native\_format** , **generic\_format** , **auto\_format** }
- using **string\_type**
- using **value\_type**

### Public Member Functions

- template<typename \_InputIterator, typename \_Require = \_\_detail::\_Path2<\_InputIterator>, typename \_Req2 = \_\_detail::\_value\_type\_<↔ is\_char<\_InputIterator>>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last, const [locale](#) &\_\_loc, **format**=auto\_format)
- template<typename \_InputIterator, typename \_Require = \_\_detail::\_Path2<\_InputIterator>>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last, **format**=auto\_format)
- template<typename \_Source, typename \_Require = \_\_detail::\_Path<\_Source>>>  
**path** (\_Source const &\_\_source, **format**=auto\_format)
- template<typename \_Source, typename \_Require = \_\_detail::\_Path<\_Source>, typename \_Require2 = \_\_detail::\_value\_type\_is\_char<↔ \_Source>>>  
**path** (\_Source const &\_\_src, const [locale](#) &\_\_loc, **format**=auto\_format)
- **path** (const [path](#) &\_\_p)=default
- **path** ([path](#) &&\_\_p) noexcept
- **path** ([string\\_type](#) &&\_\_source, **format**=auto\_format)
- template<typename \_InputIterator>  
\_\_detail::\_Path2<\_InputIterator> & **append** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **append** (\_Source const &\_\_source)
- template<typename \_InputIterator>  
\_\_detail::\_Path2<\_InputIterator> & **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **assign** (\_Source const &\_\_source)
- [path](#) & **assign** ([string\\_type](#) &&\_\_source)
- [iterator](#) **begin** () const noexcept
- const value\_type \* **c\_str** () const noexcept
- void **clear** () noexcept
- int **compare** ([basic\\_string\\_view](#)< value\_type > \_\_s) const noexcept
- int **compare** (const [path](#) &\_\_p) const noexcept
- int **compare** (const [string\\_type](#) &\_\_s) const noexcept
- int **compare** (const value\_type \*\_\_s) const noexcept
- template<typename \_InputIterator>  
\_\_detail::\_Path2<\_InputIterator> & **concat** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **concat** (\_Source const &\_\_x)
- bool **empty** () const noexcept
- [iterator](#) **end** () const noexcept
- [path](#) **extension** () const

- [path filename](#) () const
- [std::string generic\\_string](#) () const
- template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>  
[std::basic\\_string](#)<\_CharT, \_Traits, \_Allocator> **generic\_string** (const \_Allocator &\_\_a=\_Allocator()) const
- [std::u16string generic\\_u16string](#) () const
- [std::u32string generic\\_u32string](#) () const
- [std::string generic\\_u8string](#) () const
- [std::wstring generic\\_wstring](#) () const
- bool **has\_extension** () const noexcept
- bool **has\_filename** () const noexcept
- bool **has\_parent\_path** () const noexcept
- bool **has\_relative\_path** () const noexcept
- bool **has\_root\_directory** () const noexcept
- bool **has\_root\_name** () const noexcept
- bool **has\_root\_path** () const noexcept
- bool **has\_stem** () const noexcept
- bool **is\_absolute** () const noexcept
- bool **is\_relative** () const noexcept
- [path lexically\\_normal](#) () const
- [path lexically\\_proximate](#) (const [path](#) &base) const
- [path lexically\\_relative](#) (const [path](#) &base) const
- [path](#) & **make\_preferred** ()
- const [string\\_type](#) & **native** () const noexcept
- **operator string\_type** () const
- template<typename \_CharT>  
\_\_detail::\_Path2<\_CharT \* > & **operator+=** (\_CharT \_\_x)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **operator+=** (\_Source const &\_\_x)
- [path](#) & **operator+=** ([basic\\_string\\_view](#)<value\_type> \_\_x)
- [path](#) & **operator+=** (const [path](#) &\_\_x)
- [path](#) & **operator+=** (const [string\\_type](#) &\_\_x)
- [path](#) & **operator+=** (const value\_type \*\_\_x)
- [path](#) & **operator+=** (value\_type \_\_x)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **operator/=** (\_Source const &\_\_source)
- [path](#) & **operator/=** (const [path](#) &\_\_p)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **operator=** (\_Source const &\_\_source)
- [path](#) & **operator=** (const [path](#) &)
- [path](#) & **operator=** ([path](#) &&) noexcept
- [path](#) & **operator=** ([string\\_type](#) &&\_\_source)
- [path parent\\_path](#) () const
- [path relative\\_path](#) () const
- [path](#) & **remove\_filename** ()
- [path](#) & **replace\_extension** (const [path](#) &\_\_replacement=[path](#)())
- [path](#) & **replace\_filename** (const [path](#) &\_\_replacement)
- [path root\\_directory](#) () const
- [path root\\_name](#) () const
- [path root\\_path](#) () const
- [path stem](#) () const
- [std::string string](#) () const

- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > string (const _Allocator &__a) const`
- `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>`  
`std::basic_string< _CharT, _Traits, _Allocator > string (const _Allocator &__a=_Allocator()) const`
- `void swap (path &__rhs) noexcept`
- `std::u16string u16string () const`
- `std::u32string u32string () const`
- `std::string u8string () const`
- `std::wstring wstring () const`

### Static Public Attributes

- static constexpr value\_type **preferred\_separator**

### Friends

- `path operator/ (const path &__lhs, const path &__rhs)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const path &__p)`
- `strong_ordering operator<=> (const path &__lhs, const path &__rhs) noexcept`
- `bool operator== (const path &__lhs, const path &__rhs) noexcept`
- `template<typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, path &__p)`

### Related Symbols

(Note that these are not member symbols.)

- `void swap (path &__lhs, path &__rhs) noexcept`
- `template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _CharT = __detail::_value_type_↔`  
`is_char_or_char8_t<_InputIterator>>>`  
`path u8path (_InputIterator __first, _InputIterator __last)`
- `template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __detail::_value_type_is_char_or_↔`  
`_char8_t<_Source>>>`  
`path u8path (const _Source &__source)`

#### 5.783.1 Detailed Description

A filesystem path.

Since

C++17

#### 5.783.2 Member Enumeration Documentation

##### format

`enum std::filesystem::path::format : unsigned char`  
`path::format` is ignored in this implementation

### 5.783.3 Friends And Related Symbol Documentation

#### **operator/**

```
path operator/ (
    const path & __lhs,
    const path & __rhs) [friend]
```

Append one path to another.

#### **operator<<**

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const path & __p) [friend]
```

Write a path to a stream.

#### **operator<=>**

```
strong_ordering operator<=> (
    const path & __lhs,
    const path & __rhs) [friend]
```

Compare paths.

#### **operator==**

```
bool operator== (
    const path & __lhs,
    const path & __rhs) [friend]
```

Compare paths.

#### **operator>>**

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    path & __p) [friend]
```

Read a path from a stream.

References [std::move\(\)](#).

The documentation for this class was generated from the following file:

- [bits/fs\\_path.h](#)

## 5.784 `std::piecewise_constant_distribution<_RealType>` Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)



## Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW>`  
**piecewise\_constant\_distribution** (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- **piecewise\_constant\_distribution** (`const param_type &__p`)
- `template<typename _Func>`  
**piecewise\_constant\_distribution** (`initializer_list<_RealType> __bl, _Func __fw`)
- `template<typename _Func>`  
**piecewise\_constant\_distribution** (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>`  
**void \_\_generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>`  
**void \_\_generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator>`  
**void \_\_generate** (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- **std::vector< double > densities** () const
- **std::vector<\_RealType> intervals** () const
- **result\_type max** () const
- **result\_type min** () const
- `template<typename _UniformRandomNumberGenerator>`  
**result\_type operator()** (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator>`  
**result\_type operator()** (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- **param\_type param** () const
- **void param** (`const param_type &__param`)
- **void reset** ()

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits>`  
**std::basic\_ostream<\_CharT, \_Traits> & operator<<** (`std::basic_ostream<_CharT, _Traits> &__os, const std::piecewise_constant_distribution<_RealType1> &__x`)
- **bool operator==** (`const piecewise_constant_distribution &__d1, const piecewise_constant_distribution &__d2`)
- `template<typename _RealType1, typename _CharT, typename _Traits>`  
**std::basic\_istream<\_CharT, \_Traits> & operator>>** (`std::basic_istream<_CharT, _Traits> &__is, std::piecewise_constant_distribution<_RealType1> &__x`)

### 5.784.1 Detailed Description

**template<typename \_RealType = double>**  
**class std::piecewise\_constant\_distribution<\_RealType>**

A `piecewise_constant_distribution` random number distribution.

This distribution produces random numbers  $x$ ,  $b_0 \leq x < b_n$ , uniformly distributed over each subinterval  $[b_i, b_{i+1})$  according to the probability mass function

$$p(x|b_0, \dots, b_n, \rho_0, \dots, \rho_{n-1}) = \rho_i \cdot \frac{b_{i+1} - x}{b_{i+1} - b_i} + \rho_{i+1} \cdot \frac{x - b_i}{b_{i+1} - b_i}$$

for  $b_i \leq x < b_{i+1}$ .

Since

C++11

### 5.784.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
```

```
typedef _RealType std::piecewise_constant_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 5.784.3 Member Function Documentation

#### densities()

```
template<typename _RealType = double>
```

```
std::vector< double > std::piecewise_constant_distribution< _RealType >::densities () const  
[inline]
```

Returns a vector of the probability densities.

#### intervals()

```
template<typename _RealType = double>
```

```
std::vector< _RealType > std::piecewise_constant_distribution< _RealType >::intervals () const  
[inline]
```

Returns a vector of the intervals.

#### max()

```
template<typename _RealType = double>
```

```
result_type std::piecewise_constant_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
```

```
result_type std::piecewise_constant_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()()

```
template<typename _RealType = double>
```

```
template<typename _UniformRandomNumberGenerator>
```

```
result_type std::piecewise_constant_distribution< _RealType >::operator() (  
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

#### param() [1/2]

```
template<typename _RealType = double>
```

```
param_type std::piecewise_constant_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### param() [2/2]

```
template<typename _RealType = double>
```

```
void std::piecewise_constant_distribution< _RealType >::param (  
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::piecewise_constant_distribution<_RealType>::reset () [inline]
Resets the distribution state.
```

**5.784.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::piecewise_constant_distribution<_RealType1> & __x) [friend]
Inserts a piecewise_constant_distribution random number distribution __x into the output stream __os.
```

## Parameters

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <code>__os</code> | An output stream.                                             |
| <code>__x</code>  | A piecewise_constant_distribution random number distribution. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
    const piecewise_constant_distribution<_RealType> & __d1,
    const piecewise_constant_distribution<_RealType> & __d2) [friend]
Return true if two piecewise constant distributions have the same parameters.
```

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (
    std::basic_istream<_CharT, _Traits> & __is,
    std::piecewise_constant_distribution<_RealType1> & __x) [friend]
Extracts a piecewise_constant_distribution random number distribution __x from the input stream __is.
```

## Parameters

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                  |
| <code>__x</code>  | A piecewise_constant_distribution random number generator engine. |

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.785 `std::piecewise_construct_t` Struct Reference

```
#include <stl_pair.h>
```

### 5.785.1 Detailed Description

Tag type for piecewise construction of `std::pair` objects.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 5.786 `std::piecewise_linear_distribution<_RealType>` Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW>`  
**`piecewise_linear_distribution`** (`_InputIteratorB __bfirst`, `_InputIteratorB __bend`, `_InputIteratorW __wbegin`)
- **`piecewise_linear_distribution`** (const [param\\_type](#) &\_\_p)
- `template<typename _Func>`  
**`piecewise_linear_distribution`** ([initializer\\_list](#)< `_RealType` > \_\_bl, `_Func` \_\_fw)
- `template<typename _Func>`  
**`piecewise_linear_distribution`** (size\_t \_\_nw, `_RealType` \_\_xmin, `_RealType` \_\_xmax, `_Func` \_\_fw)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>`  
void **`__generate`** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>`  
void **`__generate`** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- `template<typename _UniformRandomNumberGenerator>`  
void **`__generate`** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- `std::vector< double >` **`densities`** () const
- `std::vector< _RealType >` **`intervals`** () const
- [result\\_type](#) **`max`** () const
- [result\\_type](#) **`min`** () const
- `template<typename _UniformRandomNumberGenerator>`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`)

- template<typename \_UniformRandomNumberGenerator>  
result\_type operator() (\_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- param\_type param () const
- void param (const param\_type &\_\_param)
- void reset ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const std::piecewise\_linear\_distribution< \_RealType1 > &\_\_x)
- bool operator== (const piecewise\_linear\_distribution &\_\_d1, const piecewise\_linear\_distribution &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
std::basic\_istream< \_CharT, \_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is, std::piecewise\_linear\_distribution< \_RealType1 > &\_\_x)

## 5.786.1 Detailed Description

template<typename \_RealType = double>  
class std::piecewise\_linear\_distribution<\_RealType>

A piecewise\_linear\_distribution random number distribution.

This distribution produces random numbers  $x$ ,  $b_0 \leq x < b_n$ , distributed over each subinterval  $[b_i, b_{i+1})$  according to the probability mass function  $p(x|b_0, \dots, b_n, \rho_0, \dots, \rho_n) = \rho_i$ , for  $b_i \leq x < b_{i+1}$ .

Since

C++11

## 5.786.2 Member Typedef Documentation

### result\_type

template<typename \_RealType = double>  
typedef \_RealType std::piecewise\_linear\_distribution<\_RealType>::result\_type  
The type of the range of the distribution.

## 5.786.3 Member Function Documentation

### densities()

template<typename \_RealType = double>  
std::vector< double > std::piecewise\_linear\_distribution<\_RealType>::densities () const [inline]  
Return a vector of the probability densities of the distribution.

### intervals()

template<typename \_RealType = double>  
std::vector< \_RealType > std::piecewise\_linear\_distribution<\_RealType>::intervals () const [inline]  
Return the intervals of the distribution.

### max()

template<typename \_RealType = double>  
result\_type std::piecewise\_linear\_distribution<\_RealType>::max () const [inline]  
Returns the least upper bound value of the distribution.

**min()**

```
template<typename _RealType = double>
result_type std::piecewise_linear_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

**operator()()**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::piecewise_linear_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

**param() [1/2]**

```
template<typename _RealType = double>
param_type std::piecewise_linear_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _RealType = double>
void std::piecewise_linear_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::piecewise_linear_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

**5.786.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::piecewise_linear_distribution< _RealType1 > & __x) [friend]
```

Inserts a `piecewise_linear_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                                          |
|-------------------|--------------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                        |
| <code>__x</code>  | A <code>piecewise_linear_distribution</code> random number distribution. |

**Returns**

The output stream with the state of \_\_\_x inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
    const piecewise\_linear\_distribution< _RealType > & __d1,
    const piecewise\_linear\_distribution< _RealType > & __d2) [friend]
```

Return true if two piecewise linear distributions have the same parameters.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::piecewise_linear_distribution< _RealType1 > & __x) [friend]
```

Extracts a `piecewise_linear_distribution` random number distribution \_\_\_x from the input stream \_\_is.

**Parameters**

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                             |
| <code>__x</code>  | A <code>piecewise_linear_distribution</code> random number generator engine. |

**Returns**

The input stream with \_\_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.787 std::plus<\_Tp> Struct Template Reference**

```
#include <stl_function.h>
```



Inheritance diagram for `std::plus<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `_Tp` [operator\(\)](#) (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

#### 5.787.1 Detailed Description

template<typename `_Tp`>

struct `std::plus<_Tp>`

One of the [math functors](#).

#### 5.787.2 Member Typedef Documentation

##### `first_argument_type`

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Tp`, `_Tp` >::`first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

##### `result_type`

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Tp`, `_Tp` >::`result_type` [inherited]

`result_type` is the return type

##### `second_argument_type`

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Tp`, `_Tp` >::`second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

### 5.787.3 Member Function Documentation

#### operator>()

```
template<typename _Tp>
_Tp std::plus< _Tp >::operator() (
    const _Tp & __x,
    const _Tp & __y) const [inline], [constexpr]
```

Returns the sum.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.788 \_\_gnu\_pbds::point\_invalidation\_guarantee Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::point\_invalidation\_guarantee:



### 5.788.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased, regardless of modifications to the container object.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.789 std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::pointer_to_binary_function<_Arg1, _Arg2, _Result>`:



### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- `pointer_to_binary_function` (`_Result(*__x)(_Arg1, _Arg2)`)
- `_Result operator()` (`_Arg1 __x, _Arg2 __y`) const

### Protected Attributes

- `_Result(*_M_ptr)(_Arg1, _Arg2)`

#### 5.789.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
class std::pointer_to_binary_function<_Arg1, _Arg2, _Result>
```

One of the [adaptors for function pointers](#).

#### 5.789.2 Member Typedef Documentation

##### `first_argument_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary\_function<\_Arg1, \_Arg2, \_Result>::first\_argument\_type [inherited]
first\_argument\_type is the type of the first argument
```

##### `result_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary\_function<\_Arg1, \_Arg2, \_Result>::result\_type [inherited]
result\_type is the return type
```

**second\_argument\_type**

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary\_function< _Arg1, _Arg2, _Result >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this class was generated from the following file:
```

- [stl\\_function.h](#)

**5.790 `std::pointer_to_unary_function< _Arg, _Result >` Class Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for `std::pointer_to_unary_function< _Arg, _Result >`:

**Public Types**

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

**Public Member Functions**

- **pointer\_to\_unary\_function** (`_Result`(\*\_\_x)(`_Arg`))
- `_Result` **operator()** (`_Arg` \_\_x) const

**Protected Attributes**

- `_Result`(\* **M\_ptr**)(`_Arg`)

**5.790.1 Detailed Description**

```
template<typename _Arg, typename _Result>
class std::pointer_to_unary_function< _Arg, _Result >
```

One of the [adaptors for function pointers](#).

### 5.790.2 Member Typedef Documentation

#### argument\_type

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary_function< _Arg, _Result >::argument_type [inherited]
argument_type is the type of the argument
```

#### result\_type

```
template<typename _Arg, typename _Result>
typedef _Result std::unary_function< _Arg, _Result >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 5.791 std::pointer\_traits<\_Ptr> Struct Template Reference

```
#include <memory>
```

#### Public Types

- using [difference\\_type](#)
- using [element\\_type](#)
- using [pointer](#)
- template<typename \_Up>  
using [rebind](#)

#### Static Public Member Functions

- static pointer [pointer\\_to](#) (element\_type &\_\_r)

#### 5.791.1 Detailed Description

```
template<typename _Ptr>
struct std::pointer_traits<_Ptr>
```

Uniform interface to all pointer-like types.

Since

C++11

### 5.791.2 Member Typedef Documentation

#### difference\_type

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::difference_type [inherited]
The type used to represent the difference between two pointers.
```

#### element\_type

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::element_type [inherited]
The type pointed to.
```

**pointer**

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl<_Ptr, _Elt>::pointer [inherited]
The pointer type.
```

**rebind**

```
template<typename _Ptr, typename _Elt>
template<typename _Up>
using std::__ptr_traits_impl<_Ptr, _Elt>::rebind [inherited]
A pointer to a different type.
```

**5.791.3 Member Function Documentation****pointer\_to()**

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
static pointer std::__ptr_traits_ptr_to<_Ptr, _Elt, bool>::pointer_to (
    element_type & __r) [inline], [static], [inherited]
```

Obtain a pointer to an object.

**Parameters**

|          |                                                            |
|----------|------------------------------------------------------------|
| ↔        | A reference to an object of type <code>element_type</code> |
| ↔        |                                                            |
| ↔        |                                                            |
| ↔        |                                                            |
| <i>r</i> |                                                            |

**Returns**

```
pointer::pointer_to(__r)
```

**Precondition**

`pointer::pointer_to(__r)` is a valid expression.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

**5.792 std::pointer\_traits<\_Tp\*> Struct Template Reference**

```
#include <memory>
```

**Public Types**

- using [difference\\_type](#)
- typedef ptrdiff\_t [difference\\_type](#)
- using [element\\_type](#)
- typedef \_Tp [element\\_type](#)
- using [pointer](#)
- typedef \_Tp\* [pointer](#)
- using [rebind](#)
- template<typename \_Up>
 using [rebind](#)

### Static Public Member Functions

- static [pointer\\_pointer\\_to](#) ([element\\_type](#) &\_\_r)
- static [pointer\\_pointer\\_to](#) ([element\\_type](#) &\_\_r)
- static [pointer\\_pointer\\_to](#) ([element\\_type](#) &\_\_r)

#### 5.792.1 Detailed Description

```
template<typename _Tp>
struct std::pointer_traits<_Tp * >
```

Partial specialization for built-in pointers.

Since

C++11

#### 5.792.2 Member Typedef Documentation

##### **difference\_type** [1/2]

```
using std::__ptr_traits_impl<_Tp *, _Elt >::difference_type
```

The type used to represent the difference between two pointers.

##### **difference\_type** [2/2]

```
template<typename _Tp>
typedef ptrdiff_t std::pointer\_traits<_Tp * >::difference_type
```

Type used to represent the difference between two pointers.

##### **element\_type** [1/2]

```
using std::__ptr_traits_impl<_Tp *, _Elt >::element_type
```

The type pointed to.

##### **element\_type** [2/2]

```
template<typename _Tp>
typedef _Tp std::pointer\_traits<_Tp * >::element_type
```

The type pointed to.

##### **pointer** [1/2]

```
using std::__ptr_traits_impl<_Tp *, _Elt >::pointer
```

The pointer type.

##### **pointer** [2/2]

```
template<typename _Tp>
typedef _Tp* std::pointer\_traits<_Tp * >::pointer
```

The pointer type.

##### **rebind** [1/2]

```
using std::__ptr_traits_impl<_Tp *, _Elt >::rebind
```

A pointer to a different type.

**rebind** [2/2]

```
template<typename _Tp>
template<typename _Up>
using std::pointer_traits< _Tp * >::rebind
A pointer to a different type.
```

**5.792.3 Member Function Documentation****pointer\_to()** [1/3]

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
static pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
    element_type & __r) [inline], [static], [inherited]
```

Obtain a pointer to an object.

**Parameters**

|          |                                               |
|----------|-----------------------------------------------|
| ↔        | A reference to an object of type element_type |
| ↔        |                                               |
| ↔        |                                               |
| ↔        |                                               |
| <i>r</i> |                                               |

**Returns**

pointer::pointer\_to(\_\_r)

**Precondition**

pointer::pointer\_to(\_\_r) is a valid expression.

**pointer\_to()** [2/3]

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
static pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
    element_type & __r) [inline], [static], [inherited]
```

Obtain a pointer to an object.

**Parameters**

|          |                                               |
|----------|-----------------------------------------------|
| ↔        | A reference to an object of type element_type |
| ↔        |                                               |
| ↔        |                                               |
| ↔        |                                               |
| <i>r</i> |                                               |

**Returns**

pointer::pointer\_to(\_\_r)

**Precondition**

pointer::pointer\_to(\_\_r) is a valid expression.



**pointer\_to()** [3/3]

```
static pointer std::__ptr_traits_ptr_to< _Tp *, _Elt, bool >::pointer_to (
    element\_type & __r) [inline], [static]
```

Obtain a pointer to an object.

**Parameters**

|          |                                                            |
|----------|------------------------------------------------------------|
| ↔        | A reference to an object of type <code>element_type</code> |
| ↔        |                                                            |
| ↔        |                                                            |
| ↔        |                                                            |
| <i>r</i> |                                                            |

**Returns**

```
pointer::pointer_to(__r)
```

**Precondition**

`pointer::pointer_to(__r)` is a valid expression.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

**5.793 std::poisson\_distribution< \_IntType > Class Template Reference**

```
#include <random>
```

**Classes**

- struct [param\\_type](#)

**Public Types**

- typedef `_IntType` [result\\_type](#)

**Public Member Functions**

- **poisson\_distribution** (const [param\\_type](#) &\_\_p)
- **poisson\_distribution** (double \_\_mean)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- double **mean** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)

- template<typename \_UniformRandomNumberGenerator>  
result\_type operator() (\_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- param\_type param () const
- void param (const param\_type &\_\_param)
- void reset ()

## Friends

- template<typename \_IntType1, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const std::poisson\_distribution< \_IntType1 > &\_\_x)
- bool operator== (const poisson\_distribution &\_\_d1, const poisson\_distribution &\_\_d2)
- template<typename \_IntType1, typename \_CharT, typename \_Traits>  
std::basic\_istream< \_CharT, \_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is, std::poisson\_distribution< \_IntType1 > &\_\_x)

### 5.793.1 Detailed Description

template<typename \_IntType = int>  
class std::poisson\_distribution< \_IntType >

A discrete Poisson random number distribution.

The formula for the Poisson probability density function is  $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$  where  $\mu$  is the parameter of the distribution.

Since

C++11

### 5.793.2 Member Typedef Documentation

#### result\_type

```
template<typename _IntType = int>
typedef _IntType std::poisson_distribution< _IntType >::result_type
```

The type of the range of the distribution.

### 5.793.3 Member Function Documentation

#### max()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### mean()

```
template<typename _IntType = int>
double std::poisson_distribution< _IntType >::mean () const [inline]
```

Returns the distribution parameter mean.

#### min()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

**operator>() [1/2]**

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::poisson_distribution< _IntType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

**operator()() [2/2]**

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
poisson_distribution< _IntType >::result_type std::poisson_distribution< _IntType >::operator()
(
    _UniformRandomNumberGenerator & __urng,
    const param_type & __param)
```

A rejection algorithm when mean  $\geq 12$  and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_FUNCS` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

References [std::abs\(\)](#), [std::numeric\\_limits< \\_Tp >::epsilon\(\)](#), [std::log\(\)](#), and [std::numeric\\_limits< \\_Tp >::max\(\)](#).

**param() [1/2]**

```
template<typename _IntType = int>
param_type std::poisson_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

**5.793.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::poisson_distribution< _IntType1 > & __x) [friend]
```

Inserts a `poisson_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| <code>__os</code> | An output stream.                                               |
| <code>__x</code>  | A <code>poisson_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### `operator==`

```
template<typename _IntType = int>
bool operator== (
    const poisson_distribution< _IntType > & __d1,
    const poisson_distribution< _IntType > & __d2) [friend]
```

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

#### `operator>>`

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::poisson_distribution< _IntType1 > & __x) [friend]
```

Extracts a `poisson_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                    |
| <code>__x</code>  | A <code>poisson_distribution</code> random number generator engine. |

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.794 `std::pmr::polymorphic_allocator<_Tp>` Class Template Reference

```
#include <memory_resource>
```

#### Public Types

- using `value_type`

## Public Member Functions

- **polymorphic\_allocator** (const [polymorphic\\_allocator](#) &\_\_other)=default
- template<typename \_Up>  
**polymorphic\_allocator** (const [polymorphic\\_allocator](#)< \_Up > &\_\_x) noexcept
- **polymorphic\_allocator** ([memory\\_resource](#) \*\_\_r) noexcept
- \_Tp \* **allocate** (size\_t \_\_n)
- template<typename \_Tp1, typename... \_Args>  
void **construct** (\_Tp1 \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (\_Tp \*\_\_p, size\_t \_\_n) noexcept
- template<typename \_Up>  
void **destroy** (\_Up \*\_\_p)
- [polymorphic\\_allocator](#) & **operator=** (const [polymorphic\\_allocator](#) &)=delete
- [memory\\_resource](#) \* **resource** () const noexcept
- [polymorphic\\_allocator](#) **select\_on\_container\_copy\_construction** () const noexcept

## Friends

- bool **operator==** (const [polymorphic\\_allocator](#) &\_\_a, const [polymorphic\\_allocator](#) &\_\_b) noexcept

### 5.794.1 Detailed Description

```
template<typename _Tp>
class std::pmr::polymorphic_allocator< _Tp >
```

Class template polymorphic\_allocator.

Since

C++17

The documentation for this class was generated from the following file:

- [memory\\_resource.h](#)

## 5.795 std::pmr::pool\_options Struct Reference

```
#include <memory_resource>
```

## Public Attributes

- size\_t **largest\_required\_pool\_block**
- size\_t [max\\_blocks\\_per\\_chunk](#)

### 5.795.1 Detailed Description

Parameters for tuning a pool resource's behaviour.

Since

C++17

### 5.795.2 Member Data Documentation

#### `max_blocks_per_chunk`

`size_t std::pmr::pool_options::max_blocks_per_chunk`

Upper limit on number of blocks in a chunk.

A lower value prevents allocating huge chunks that could remain mostly unused, but means pools will need to replenished more frequently.

The documentation for this struct was generated from the following file:

- [memory\\_resource](#)

## 5.796 `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>` Class Template Reference

```
#include <priority_queue.hpp>
```

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `Tag` **container\_category**
- typedef `allocator_type::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `allocator_type::size_type` **size\_type**
- typedef `_Tv` **value\_type**

### Public Member Functions

- [priority\\_queue](#) (`const cmp_fn &r_cmp_fn`)
- [priority\\_queue](#) (`const priority\_queue &other`)
- `template<typename It>`  
[priority\\_queue](#) (`It first_it, It last_it`)
- `template<typename It>`  
[priority\\_queue](#) (`It first_it, It last_it, const cmp_fn &r_cmp_fn`)
- [priority\\_queue](#) & **operator=** (`const priority\_queue &other`)
- `void` **swap** ([priority\\_queue](#) &other)

### 5.796.1 Detailed Description

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
```

```
class __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>
```

A priority queue composed of one specific heap policy.

## Template Parameters

|                     |                                                                     |
|---------------------|---------------------------------------------------------------------|
| <code>_Tv</code>    | Value type.                                                         |
| <code>Cmp_Fn</code> | Comparison functor.                                                 |
| <code>Tag</code>    | Instantiating data structure type, see <code>container_tag</code> . |
| <code>_Alloc</code> | Allocator type.                                                     |

Base is dispatched at compile time via `Tag`, from the following choices: `binary_heap_tag`, `binomial_heap_tag`, `pairing_heap_tag`, `rc_binomial_heap_tag`, `thin_heap_tag`

Base choices are: `detail::binary_heap`, `detail::binomial_heap`, `detail::pairing_heap`, `detail::rc_binomial_heap`, `detail::thin_heap`.

## 5.796.2 Constructor & Destructor Documentation

### `priority_queue()` [1/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
    const cmp_fn & r_cmp_fn) [inline]
```

Constructor taking some policy objects. `r_cmp_fn` will be copied by the `Cmp_Fn` object of the container object.

### `priority_queue()` [2/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
    It first_it,
    It last_it) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

### `priority_queue()` [3/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
    It first_it,
    It last_it,
    const cmp_fn & r_cmp_fn) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_cmp_fn` will be copied by the `cmp_fn` object of the container object.

The documentation for this class was generated from the following file:

- [priority\\_queue.hpp](#)

## 5.797 `std::priority_queue< _Tp, _Sequence, _Compare >` Class Template Reference

```
#include <stl_queue.h>
```

### Public Types

- `typedef _Sequence::const_reference` **const\_reference**

- typedef `_Sequence` **container\_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Sequence::value_type` **value\_type**

## Public Member Functions

- template<typename `_Seq` = `_Sequence`, typename `_Requires` = `typename enable_if<__and<is_default_constructible<_Compare>, is_default_constructible<_Seq>>>::value>::type`>  
`priority_queue` ()
- template<typename `_InputIterator`, typename `_Alloc`, typename = `std::_RequireInputIter<_InputIterator>`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Alloc` &\_\_alloc)
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
`priority_queue` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x, `_Sequence` &&\_\_s)
- template<typename `_InputIterator`, typename `_Alloc`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x, `_Sequence` &&\_\_s, const `_Alloc` &\_\_alloc)
- template<typename `_InputIterator`, typename `_Alloc`, typename = `std::_RequireInputIter<_InputIterator>`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x, const `_Alloc` &\_\_alloc)
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
`priority_queue` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x, const `_Sequence` &\_\_s)
- template<typename `_InputIterator`, typename `_Alloc`, typename = `std::_RequireInputIter<_InputIterator>`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x, const `_Sequence` &\_\_s, const `_Alloc` &\_\_alloc)
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
`priority_queue` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_x=`_Compare`())
- template<typename `_Alloc`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (const `_Alloc` &\_\_a)
- template<typename `_Alloc`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (const `_Compare` &\_\_x, `_Sequence` &&\_\_c, const `_Alloc` &\_\_a)
- `priority_queue` (const `_Compare` &\_\_x, `_Sequence` &&\_\_s=`_Sequence`())
- template<typename `_Alloc`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (const `_Compare` &\_\_x, const `_Alloc` &\_\_a)
- template<typename `_Alloc`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (const `_Compare` &\_\_x, const `_Sequence` &\_\_c, const `_Alloc` &\_\_a)
- `priority_queue` (const `_Compare` &\_\_x, const `_Sequence` &\_\_s)
- `priority_queue` (const `priority_queue` &)=default
- template<typename `_Alloc`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (const `priority_queue` &\_\_q, const `_Alloc` &\_\_a)
- `priority_queue` (`priority_queue` &&\_\_q) noexcept(\_\_and<`is_nothrow_move_constructible`< `_Sequence` >, `is_nothrow_move_constructible`< `_Compare` > >::value)
- template<typename `_Alloc`, typename `_Requires` = `_Uses<_Alloc>`>>  
`priority_queue` (`priority_queue` &&\_\_q, const `_Alloc` &\_\_a)
- template<typename... `_Args`>  
void **emplace** (`_Args` &&... \_\_args)
- bool **empty** () const
- `priority_queue` & **operator=** (const `priority_queue` &)=default
- `priority_queue` & **operator=** (`priority_queue` &&\_\_q) noexcept(\_\_and<`is_nothrow_move_assignable`< `_Sequence` >, `is_nothrow_move_assignable`< `_Compare` > >::value)



- void `pop()`
- void `push(const value_type &__x)`
- void `push(value_type &&__x)`
- size\_type `size()` const
- void `swap(priority_queue &__pq)` noexcept(`__and_<__c++1z or gnu++11 __is_nothrow_swappable<_Sequence>, __is_nothrow_swappable<_Compare>>::value`)
- const\_reference `top()` const

### Protected Attributes

- `_Sequence` `c`
- `_Compare` `comp`

### 5.797.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
```

```
class std::priority_queue< _Tp, _Sequence, _Compare >
```

A standard container automatically sorting its contents.

#### Template Parameters

|                        |                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                                                              |
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>vector&lt;_Tp&gt;</code> .                     |
| <code>_Compare</code>  | Comparison function object type, defaults to <code>less&lt;_Sequence::value_type&gt;</code> . |

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push`, `pop`, and `top`, which are standard queue operations.

#### Note

No equality/comparison operators are provided for `priority_queue`.

Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

### 5.797.2 Constructor & Destructor Documentation

#### `priority_queue()` [1/2]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
```

```
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<__and_<is_default_constructible<_Compare>, is_default_constructible<_Seq>>::value>::type>
```

```
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue () [inline]
```

Default constructor creates no elements.

**priority\_queue()** [2/2]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
    _InputIterator __first,
    _InputIterator __last,
    const _Compare & __x = _Compare()) [inline]
```

Builds a queue from a range.

**Parameters**

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| <code>__first</code> | An input iterator.                                      |
| <code>__last</code>  | An input iterator.                                      |
| <code>__x</code>     | A comparison functor describing a strict weak ordering. |
| <code>__s</code>     | An initial sequence with which to start.                |

Begins by copying `__s`, inserting a copy of the elements from `[first,last)` into the copy of `__s`, then ordering the copy according to `__x`.

For more information on function objects, see the documentation on [functor base classes](#).

**5.797.3 Member Function Documentation****empty()**

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>>
bool std::priority_queue< _Tp, _Sequence, _Compare >::empty () const [inline], [nodiscard]
```

Returns true if the queue is empty.

Referenced by [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#).

**pop()**

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>>
void std::priority_queue< _Tp, _Sequence, _Compare >::pop () [inline]
```

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Referenced by [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#).

**push()**

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>>
void std::priority_queue< _Tp, _Sequence, _Compare >::push (
    const value_type & __x) [inline]
```

Add data to the queue.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__↵</code> | Data to be added. |
| <code>__x</code> |                   |

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence. Referenced by [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#).

### size()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
size_type std::priority\_queue< _Tp, _Sequence, _Compare >::size () const [inline], [nodiscard]
```

Returns the number of elements in the queue.

### top()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
const_reference std::priority\_queue< _Tp, _Sequence, _Compare >::top () const [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the first element of the queue. Referenced by [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#). The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 5.798 \_\_gnu\_pbds::priority\_queue\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::priority_queue_tag`:



### 5.798.1 Detailed Description

Basic priority-queue.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.799 `__gnu_pbds::detail::probe_fn_base<_Alloc>` Class Template Reference

```
#include <probe_fn_base.hpp>
```

### 5.799.1 Detailed Description

```
template<typename _Alloc>
class __gnu_pbds::detail::probe_fn_base<_Alloc>
```

Probe functor base.

The documentation for this class was generated from the following file:

- [probe\\_fn\\_base.hpp](#)

## 5.800 `__gnu_cxx::project1st<_Arg1, _Arg2>` Struct Template Reference

```
#include <functional>
```

### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Arg1` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- `_Arg1` [operator\(\)](#) (const `_Arg1` &`_x`, const `_Arg2` &) const

### 5.800.1 Detailed Description

```
template<class _Arg1, class _Arg2>
struct __gnu_cxx::project1st<_Arg1, _Arg2>
```

An [SGI extension](#).

### 5.800.2 Member Typedef Documentation

#### `first_argument_type`

```
typedef _Arg1 std::binary\_function<\_Arg1, \_Arg2, \_Arg1>::first\_argument\_type [inherited]
first_argument_type is the type of the first argument
```

#### `result_type`

```
typedef _Arg1 std::binary\_function<\_Arg1, \_Arg2, \_Arg1>::result\_type [inherited]
result_type is the return type
```

#### `second_argument_type`

```
typedef _Arg2 std::binary\_function<\_Arg1, \_Arg2, \_Arg1>::second\_argument\_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.801 `__gnu_cxx::project2nd<_Arg1, _Arg2>` Struct Template Reference

```
#include <functional>
```

## Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Arg2` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Arg2 operator()` (const `_Arg1` &, const `_Arg2` &\_\_y) const

### 5.801.1 Detailed Description

```
template<class _Arg1, class _Arg2>
struct __gnu_cxx::project2nd< _Arg1, _Arg2 >
```

An [SGI extension](#) .

### 5.801.2 Member Typedef Documentation

#### `first_argument_type`

```
typedef _Arg1 std::binary\_function< _Arg1, _Arg2, _Arg2 >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### `result_type`

```
typedef _Arg2 std::binary\_function< _Arg1, _Arg2, _Arg2 >::result_type [inherited]
result_type is the return type
```

#### `second_argument_type`

```
typedef _Arg2 std::binary\_function< _Arg1, _Arg2, _Arg2 >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.802 `std::promise< _Res >` Class Template Reference

```
#include <future>
```

## Public Member Functions

- template<typename `_Allocator`>  
`promise` (allocator\_arg\_t, const `_Allocator` &, [promise](#) &&\_\_rhs)
- template<typename `_Allocator`>  
`promise` (allocator\_arg\_t, const `_Allocator` &\_\_a)
- `promise` (const [promise](#) &)=delete
- `promise` ([promise](#) &&\_\_rhs) noexcept
- `future< _Res >` `get_future` ()
- `promise` & `operator=` (const [promise](#) &)=delete
- `promise` & `operator=` ([promise](#) &&\_\_rhs) noexcept
- void `set_exception` ([exception\\_ptr](#) \_\_p)
- void `set_exception_at_thread_exit` ([exception\\_ptr](#) \_\_p)
- void `set_value` (`_Res` &&\_\_r)
- void `set_value` (const `_Res` &\_\_r)

- void **set\_value\_at\_thread\_exit** (\_Res &&\_\_r)
- void **set\_value\_at\_thread\_exit** (const \_Res &\_\_r)
- void **swap** (promise &\_\_rhs) noexcept

### Friends

- template<typename, typename>  
struct \_State::\_Setter

### 5.802.1 Detailed Description

template<typename \_Res>  
class std::promise<\_Res >

Primary template for promise.

The documentation for this class was generated from the following file:

- [future](#)

## 5.803 std::promise<\_Res & > Class Template Reference

```
#include <future>
```

### Public Member Functions

- **promise** (allocator\_arg\_t, const \_Allocator &, [promise](#) &&\_\_rhs)
- template<typename \_Allocator>  
**promise** (allocator\_arg\_t, const \_Allocator &, [promise](#) &&\_\_rhs)
- **promise** (allocator\_arg\_t, const \_Allocator &\_\_a)
- template<typename \_Allocator>  
**promise** (allocator\_arg\_t, const \_Allocator &\_\_a)
- **promise** (const [promise](#) &)=delete
- **promise** (const [promise](#) &)=delete
- **promise** ([promise](#) &&\_\_rhs) noexcept
- **promise** ([promise](#) &&\_\_rhs) noexcept
- [future](#)<\_Res > **get\_future** ()
- [future](#)<\_Res & > **get\_future** ()
- [promise](#) & **operator=** (const [promise](#) &)=delete
- [promise](#) & **operator=** (const [promise](#) &)=delete
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- void **set\_exception** (exception\_ptr \_\_p)
- void **set\_exception** (exception\_ptr \_\_p)
- void **set\_exception\_at\_thread\_exit** (exception\_ptr \_\_p)
- void **set\_exception\_at\_thread\_exit** (exception\_ptr \_\_p)
- void **set\_value** (\_Res &&\_\_r)
- void **set\_value** (\_Res &\_\_r)
- void **set\_value** (const \_Res &\_\_r)
- void **set\_value\_at\_thread\_exit** (\_Res &&\_\_r)
- void **set\_value\_at\_thread\_exit** (\_Res &\_\_r)
- void **set\_value\_at\_thread\_exit** (const \_Res &\_\_r)
- void **swap** ([promise](#) &\_\_rhs) noexcept
- void **swap** ([promise](#) &\_\_rhs) noexcept

## Friends

- `struct _State::_Setter`
- `template<typename, typename>`  
`struct _State::_Setter`

### 5.803.1 Detailed Description

```
template<typename _Res>
class std::promise<_Res & >
```

Partial specialization for `promise<R&>`

The documentation for this class was generated from the following file:

- [future](#)

## 5.804 `std::promise< void >` Class Reference

```
#include <future>
```

### Public Member Functions

- **promise** (`allocator_arg_t`, `const _Allocator &`, [promise](#) &&\_\_rhs)
- `template<typename _Allocator>`  
**promise** (`allocator_arg_t`, `const _Allocator &`, [promise](#) &&\_\_rhs)
- **promise** (`allocator_arg_t`, `const _Allocator &` \_\_a)
- `template<typename _Allocator>`  
**promise** (`allocator_arg_t`, `const _Allocator &` \_\_a)
- **promise** (`const promise &`)=delete
- **promise** (`const promise &`)=delete
- **promise** ([promise](#) &&\_\_rhs) noexcept
- **promise** ([promise](#) &&\_\_rhs) noexcept
- [future](#)< void > **get\_future** ()
- [future](#)< void > **get\_future** ()
- [promise](#) & **operator=** (`const promise &`)=delete
- [promise](#) & **operator=** (`const promise &`)=delete
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- void **set\_exception** (`exception_ptr` \_\_p)
- void **set\_exception** (`exception_ptr` \_\_p)
- void **set\_exception\_at\_thread\_exit** (`exception_ptr` \_\_p)
- void **set\_exception\_at\_thread\_exit** (`exception_ptr` \_\_p)
- void **set\_value** ()
- void **set\_value** (`const void &` \_\_r)
- void **set\_value** (`void &&` \_\_r)
- void **set\_value\_at\_thread\_exit** ()
- void **set\_value\_at\_thread\_exit** (`const void &` \_\_r)
- void **set\_value\_at\_thread\_exit** (`void &&` \_\_r)
- void **swap** ([promise](#) &\_\_rhs) noexcept
- void **swap** ([promise](#) &\_\_rhs) noexcept

## Friends

- struct **\_State::Setter**
- template<typename, typename>  
struct **\_State::Setter**

### 5.804.1 Detailed Description

Explicit specialization for promise<void>

The documentation for this class was generated from the following file:

- [future](#)

## 5.805 std::experimental::fundamentals\_v2::propagate\_const<\_Tp> Class Template Reference

```
#include <propagate_const>
```

## Public Types

- using **element\_type**

## Public Member Functions

- template<typename \_Up, typename [enable\\_if](#)< \_\_and< [is\\_constructible](#)< \_Tp, \_Up && >, [is\\_convertible](#)< \_Up &&, \_Tp >, \_\_not\_< \_\_is\_propagate\_const< typename [decay](#)< \_Up >::type > >::value, bool >::type = true>  
constexpr **propagate\_const** (\_Up &&\_\_u)
- template<typename \_Up, typename [enable\\_if](#)< \_\_and< [is\\_constructible](#)< \_Tp, \_Up && >, \_\_not\_< [is\\_convertible](#)< \_Up &&, \_Tp > >, \_\_not\_< \_\_is\_propagate\_const< typename [decay](#)< \_Up >::type > >::value, bool >::type = false>  
constexpr **propagate\_const** (\_Up &&\_\_u)
- **propagate\_const** (const [propagate\\_const](#) &\_\_p)=delete
- constexpr **propagate\_const** ([propagate\\_const](#) &&\_\_p)=default
- template<typename \_Up, typename [enable\\_if](#)< \_\_and< [is\\_constructible](#)< \_Tp, \_Up && >, [is\\_convertible](#)< \_Up &&, \_Tp > >::value, bool >::type = true>  
constexpr **propagate\_const** ([propagate\\_const](#)< \_Up > &&\_\_pu)
- template<typename \_Up, typename [enable\\_if](#)< \_\_and< [is\\_constructible](#)< \_Tp, \_Up && >, \_\_not\_< [is\\_convertible](#)< \_Up &&, \_Tp > > >::value, bool >::type = false>  
constexpr **propagate\_const** ([propagate\\_const](#)< \_Up > &&\_\_pu)
- constexpr element\_type \* **get** ()
- constexpr const element\_type \* **get** () const
- constexpr **operator bool** () const
- constexpr element\_type & **operator\*** ()
- constexpr const element\_type & **operator\*** () const
- constexpr element\_type \* **operator->** ()
- constexpr const element\_type \* **operator->** () const
- template<typename \_Up, typename = typename [enable\\_if](#)< \_\_and< [is\\_convertible](#)< \_Up&&, \_Tp>, \_\_not\_< \_\_is\_propagate\_const< type-name [decay](#)< \_Up >::type > > >::value >::type >  
constexpr [propagate\\_const](#) & **operator=** (\_Up &&\_\_u)
- [propagate\\_const](#) & **operator=** (const [propagate\\_const](#) &\_\_p)=delete
- constexpr [propagate\\_const](#) & **operator=** ([propagate\\_const](#) &&\_\_p)=default
- template<typename \_Up, typename = typename [enable\\_if](#)< [is\\_convertible](#)< \_Up&&, \_Tp>::value >::type >  
constexpr [propagate\\_const](#) & **operator=** ([propagate\\_const](#)< \_Up > &&\_\_pu)
- constexpr void **swap** ([propagate\\_const](#) &\_\_pt) noexcept(\_\_is\_nothrow\_swappable< \_Tp >::value)



## Friends

- `template<typename _Up>`  
`constexpr const _Up & get_underlying (const propagate\_const< _Up > &__pt) noexcept`
- `template<typename _Up>`  
`constexpr _Up & get_underlying (propagate\_const< _Up > &__pt) noexcept`

### 5.805.1 Detailed Description

`template<typename _Tp>`  
`class std::experimental::fundamentals_v2::propagate_const< _Tp >`

Const-propagating wrapper.

The documentation for this class was generated from the following file:

- [propagate\\_const](#)

## 5.806 `__gnu_pbds::quadratic_probe_fn< Size_Type >` Class Template Reference

```
#include <hash_policy.hpp>
```

### Public Types

- `typedef Size_Type size_type`

### Public Member Functions

- `void swap (quadratic\_probe\_fn< Size_Type > &other)`

### Protected Member Functions

- `size_type operator\(\) (size_type i) const`

### 5.806.1 Detailed Description

`template<typename Size_Type = std::size_t>`  
`class __gnu_pbds::quadratic_probe_fn< Size_Type >`

A probe sequence policy using square increments.

### 5.806.2 Member Function Documentation

#### `operator()()`

```
template<typename Size_Type>
quadratic\_probe\_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator()
(
    size_type i) const [inline], [protected]
```

Returns the i-th offset from the hash value.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 5.807 `std::queue< _Tp, _Sequence >` Class Template Reference

```
#include <stl_queue.h>
```

## Public Types

- typedef \_Sequence::const\_reference **const\_reference**
- typedef \_Sequence **container\_type**
- typedef \_Sequence::reference **reference**
- typedef \_Sequence::size\_type **size\_type**
- typedef \_Sequence::value\_type **value\_type**

## Public Member Functions

- template<typename \_Seq = \_Sequence, typename \_Requires = typename enable\_if<is\_default\_constructible<\_Seq>::value>::type> **queue** ()
- **queue** (\_Sequence &&\_\_c)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>> **queue** (\_Sequence &&\_\_c, const \_Alloc &\_\_a)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>> **queue** (const \_Alloc &\_\_a)
- **queue** (const \_Sequence &\_\_c)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>> **queue** (const \_Sequence &\_\_c, const \_Alloc &\_\_a)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>> **queue** (const **queue** &\_\_q, const \_Alloc &\_\_a)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>> **queue** (**queue** &&\_\_q, const \_Alloc &\_\_a)
- reference **back** ()
- const\_reference **back** () const
- template<typename... \_Args> decltype(auto) **emplace** (\_Args &&... \_\_args)
- bool **empty** () const
- reference **front** ()
- const\_reference **front** () const
- void **pop** ()
- void **push** (const value\_type &\_\_x)
- void **push** (value\_type &&\_\_x)
- size\_type **size** () const
- void **swap** (**queue** &\_\_q) noexcept(\_\_is\_nothrow\_swappable<\_Sequence>::value)

## Protected Attributes

- \_Sequence **c**

## Friends

- template<typename \_Tp1, typename \_Seq1> bool **operator**< (const **queue**< \_Tp1, \_Seq1 > &, const **queue**< \_Tp1, \_Seq1 > &)
- template<typename \_Tp1, three\_way\_comparable \_Seq1> **compare\_three\_way\_result\_t**< \_Seq1 > **operator**<= > (const **queue**< \_Tp1, \_Seq1 > &, const **queue**< \_Tp1, \_Seq1 > &)
- template<typename \_Tp1, typename \_Seq1> bool **operator**== (const **queue**< \_Tp1, \_Seq1 > &, const **queue**< \_Tp1, \_Seq1 > &)

### 5.807.1 Detailed Description

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::queue< _Tp, _Sequence >
```

A standard container giving FIFO behavior.

#### Template Parameters

|                        |                                                                          |
|------------------------|--------------------------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                                         |
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> . |

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push` and `pop`, which are standard queue/FIFO operations.

### 5.807.2 Constructor & Destructor Documentation

#### `queue()`

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↔
_Seq>::value>::type>
std::queue< _Tp, _Sequence >::queue () [inline]
```

Default constructor creates no elements.

References [c](#).

### 5.807.3 Member Function Documentation

#### `back()` [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue< _Tp, _Sequence >::back () [inline], [nodiscard]
```

Returns a read/write reference to the data at the last element of the queue.

References [c](#).

#### `back()` [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue< _Tp, _Sequence >::back () const [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the last element of the queue.

References [c](#).

#### `empty()`

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::queue< _Tp, _Sequence >::empty () const [inline], [nodiscard]
```

Returns true if the queue is empty.

References [c](#).

**front()** [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue<_Tp, _Sequence>::front() [inline], [nodiscard]
```

Returns a read/write reference to the data at the first element of the queue.

References [c](#).

**front()** [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue<_Tp, _Sequence>::front() const [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the first element of the queue.

References [c](#).

**pop()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue<_Tp, _Sequence>::pop() [inline]
```

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

References [c](#).

**push()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue<_Tp, _Sequence>::push (
    const value_type & __x) [inline]
```

Add data to the end of the queue.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

References [c](#).

**size()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::queue<_Tp, _Sequence>::size() const [inline], [nodiscard]
```

Returns the number of elements in the queue.

References [c](#).

**5.807.4 Member Data Documentation****c**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
_Sequenece std::queue<_Tp, _Sequence>::c [protected]
```

`c` is the underlying container.

Referenced by [queue\(\)](#), [back\(\)](#), [back\(\)](#), [empty\(\)](#), [front\(\)](#), [front\(\)](#), [std::operator<\(\)](#), [std::operator==\(\)](#), [pop\(\)](#), [push\(\)](#), and [size\(\)](#).

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 5.808 \_\_gnu\_parallel::quicksort\_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for \_\_gnu\_parallel::quicksort\_tag:



### Public Member Functions

- **quicksort\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

#### 5.808.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

#### 5.808.2 Member Function Documentation

##### **\_\_get\_num\_threads()**

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

##### **set\_num\_threads()**

```
void __gnu_parallel::parallel_tag::set_num_threads (
    \_ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

##### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

---

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.809 std::random\_access\_iterator\_tag Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for std::random\_access\_iterator\_tag:



### 5.809.1 Detailed Description

Random-access iterators support a superset of bidirectional iterator operations. The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.810 \_\_gnu\_cxx::random\_condition Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::random_condition`:



## Classes

- struct [always\\_adjustor](#)
- struct [group\\_adjustor](#)
- struct [never\\_adjustor](#)

## Public Member Functions

- void **seed** (unsigned long \_\_s)

## Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

### 5.810.1 Detailed Description

Base class for random probability control and throw.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.811 std::random\_device Class Reference

```
#include <random>
```

## Public Types

- typedef unsigned int [result\\_type](#)

## Public Member Functions

- **random\_device** (const [random\\_device](#) &)=delete
- **random\_device** (const [std::string](#) &\_\_token)
- double **entropy** () const noexcept
- [result\\_type](#) **operator**() ()
- void **operator=** (const [random\\_device](#) &)=delete

## Static Public Member Functions

- static constexpr [result\\_type](#) **max** ()
- static constexpr [result\\_type](#) **min** ()

### 5.811.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Since

C++11

### 5.811.2 Member Typedef Documentation

#### result\_type

typedef unsigned int [std::random\\_device::result\\_type](#)

The type of the generated random value.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.812 std::range\_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for std::range\_error:



### Public Member Functions

- `range_error` (const char \*)
- `range_error` (const [range\\_error](#) &)=default
- `range_error` (const [string](#) &\_\_arg)
- `range_error` ([range\\_error](#) &&)=default
- [range\\_error](#) & `operator=` (const [range\\_error](#) &)=default
- [range\\_error](#) & `operator=` ([range\\_error](#) &&)=default
- virtual const char \* `what` () const noexcept

### 5.812.1 Detailed Description

Thrown to indicate range errors in internal computations.



### 5.812.2 Member Function Documentation

#### what()

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), and [std::filesystem::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 5.813 \_\_gnu\_pbds::range\_invalidation\_guarantee Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::range\_invalidation\_guarantee:



#### 5.813.1 Detailed Description

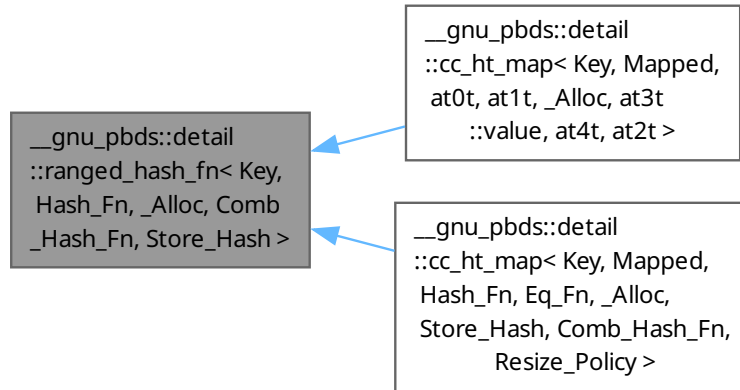
Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of `begin()` and `end()`) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not be erased, regardless of modifications to the container object.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.814 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >`:



### 5.814.1 Detailed Description

**template<typename Key, typename Hash\_Fn, typename \_Alloc, typename Comb\_Hash\_Fn, bool Store\_Hash>**  
**class** `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >`

Primary template.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

## 5.815 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >` Class Template Reference

```
#include <ranged_hash_fn.hpp>
```

### Protected Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn\_base**
- typedef Hash\_Fn **hash\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef \_Alloc::size\_type **size\_type**

### Protected Member Functions

- **ranged\_hash\_fn** (size\_type)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference) const
- void **swap** ([ranged\\_hash\\_fn](#)< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, false > &)

### 5.815.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >
```

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

### 5.816 \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, true > Class Template Reference

```
#include <ranged_hash_fn.hpp>
```

#### Protected Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn\_base**
- typedef [std::pair](#)< size\_type, size\_type > **comp\_hash**
- typedef Hash\_Fn **hash\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef \_Alloc::size\_type **size\_type**

#### Protected Member Functions

- **ranged\_hash\_fn** (size\_type)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void **notify\_resized** (size\_type)
- [comp\\_hash operator](#)() (key\_const\_reference) const
- [comp\\_hash operator](#)() (key\_const\_reference, size\_type) const
- void **swap** ([ranged\\_hash\\_fn](#)< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, true > &)

### 5.816.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
```

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

### 5.817 \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, null\_type, \_Alloc, Comb\_Hash\_Fn, false > Class Template Reference

```
#include <ranged_hash_fn.hpp>
```

#### Protected Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn\_base**
- typedef \_Alloc::size\_type **size\_type**

### Protected Member Functions

- `ranged_hash_fn` (size\_type)
- `ranged_hash_fn` (size\_type, const Comb\_Hash\_Fn &)
- `ranged_hash_fn` (size\_type, const [null\\_type](#) &, const Comb\_Hash\_Fn &)
- void `swap` ([ranged\\_hash\\_fn](#)< Key, [null\\_type](#), \_Alloc, Comb\_Hash\_Fn, false > &)

#### 5.817.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >
```

Specialization 3 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values not be stored.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

### 5.818 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

```
#include <ranged_hash_fn.hpp>
```

### Protected Types

- typedef Comb\_Hash\_Fn `comb_hash_fn_base`
- typedef \_Alloc::size\_type `size_type`

### Protected Member Functions

- `ranged_hash_fn` (size\_type)
- `ranged_hash_fn` (size\_type, const Comb\_Hash\_Fn &)
- `ranged_hash_fn` (size\_type, const [null\\_type](#) &, const Comb\_Hash\_Fn &)
- void `swap` ([ranged\\_hash\\_fn](#)< Key, [null\\_type](#), \_Alloc, Comb\_Hash\_Fn, true > &)

#### 5.818.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >
```

Specialization 4 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values be stored.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

### 5.819 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >`:



#### 5.819.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >
```

Primary template.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

### 5.820 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >` Class Template Reference

```
#include <ranged_probe_fn.hpp>
```

#### Protected Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn\_base**
- typedef Hash\_Fn **hash\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef Probe\_Fn **probe\_fn\_base**
- typedef \_Alloc::size\_type **size\_type**

#### Protected Member Functions

- **ranged\_probe\_fn** (size\_type)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &)

- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference) const
- size\_type **operator()** (key\_const\_reference, size\_type, size\_type) const
- void **swap** (ranged\_probe\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, false > &)

### 5.820.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
```

Specialization 1

The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.  
The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

## 5.821 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >` Class Template Reference

```
#include <ranged_probe_fn.hpp>
```

### Protected Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn\_base**
- typedef [std::pair](#)< size\_type, size\_type > **comp\_hash**
- typedef Hash\_Fn **hash\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef Probe\_Fn **probe\_fn\_base**
- typedef \_Alloc::size\_type **size\_type**

### Protected Member Functions

- **ranged\_probe\_fn** (size\_type)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &)
- void **notify\_resized** (size\_type)
- [comp\\_hash](#) **operator()** (key\_const\_reference) const
- size\_type **operator()** (key\_const\_reference, size\_type) const
- size\_type **operator()** (key\_const\_reference, size\_type, size\_type) const
- void **swap** (ranged\_probe\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, true > &)

### 5.821.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >
```

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

## 5.822 `__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >` Class Template Reference

```
#include <ranged_probe_fn.hpp>
```

### Protected Types

- typedef `Comb_Probe_Fn` **`comb_probe_fn_base`**
- typedef `rebind_traits< _Alloc, Key >::const_reference` **`key_const_reference`**
- typedef `_Alloc::size_type` **`size_type`**

### Protected Member Functions

- **`ranged_probe_fn`** (`size_type` `size`)
- **`ranged_probe_fn`** (`size_type`, `const Comb_Probe_Fn &r_comb_probe_fn`)
- **`ranged_probe_fn`** (`size_type`, `const null_type &`, `const Comb_Probe_Fn &r_comb_probe_fn`, `const null_type &`)
- void **`swap`** (`ranged_probe_fn &other`)

#### 5.822.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >
```

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

## 5.823 `std::rank< _Tp >` Struct Template Reference

```
#include <type_traits>
```

Inheritance diagram for `std::rank< _Tp >`:



### Public Types

- using **`type`**
- using **`value_type`**

**Public Member Functions**

- constexpr **operator value\_type** () const noexcept

**Static Public Attributes**

- static constexpr std::size\_t **value**

**5.823.1 Detailed Description**

```
template<typename _Tp>
struct std::rank<_Tp >
```

rank

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.824 `std::ratio<_Num, _Den >` Struct Template Reference**

```
#include <ratio>
```

**Public Types**

- typedef [ratio](#)< num, den > **type**

**Static Public Attributes**

- static constexpr intmax\_t **den**
- static constexpr intmax\_t **num**

**5.824.1 Detailed Description**

```
template<intmax_t _Num, intmax_t _Den = 1>
struct std::ratio<_Num, _Den >
```

Provides compile-time rational arithmetic.

This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type `intmax_t`. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

The documentation for this struct was generated from the following file:

- [ratio](#)

**5.825 `std::ratio_equal<_R1, _R2 >` Struct Template Reference**

```
#include <ratio>
```



Inheritance diagram for `std::ratio_equal<_R1, _R2>`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 5.825.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_equal<_R1, _R2>
```

ratio\_equal

The documentation for this struct was generated from the following file:

- [ratio](#)

#### 5.826 std::ratio\_greater<\_R1, \_R2> Struct Template Reference

```
#include <ratio>
```

Inheritance diagram for `std::ratio_greater<_R1, _R2>`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 5.826.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_greater<_R1, _R2>
```

`ratio_greater`

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.827 `std::ratio_greater_equal<_R1, _R2>` Struct Template Reference

```
#include <ratio>
```

Inheritance diagram for `std::ratio_greater_equal< _R1, _R2 >`:



### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr bool **value**

#### 5.827.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_greater_equal< _R1, _R2 >
```

ratio\_greater\_equal

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.828 std::ratio\_less< \_R1, \_R2 > Struct Template Reference

```
#include <ratio>
```

#### 5.828.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_less< _R1, _R2 >
```

ratio\_less

The documentation for this struct was generated from the following file:

- [ratio](#)

**5.829 `std::ratio_less_equal<_R1, _R2>` Struct Template Reference**

```
#include <ratio>
```

Inheritance diagram for `std::ratio_less_equal<_R1, _R2>`:

**Public Types**

- using **type**
- using **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** () const noexcept

**Static Public Attributes**

- static constexpr bool **value**

**5.829.1 Detailed Description**

```
template<typename _R1, typename _R2>
struct std::ratio_less_equal<_R1, _R2>
```

`ratio_less_equal`

The documentation for this struct was generated from the following file:

- [ratio](#)

**5.830 `std::ratio_not_equal<_R1, _R2>` Struct Template Reference**

```
#include <ratio>
```

Inheritance diagram for `std::ratio_not_equal<_R1, _R2>`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 5.830.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_not_equal<_R1, _R2>
```

`ratio_not_equal`

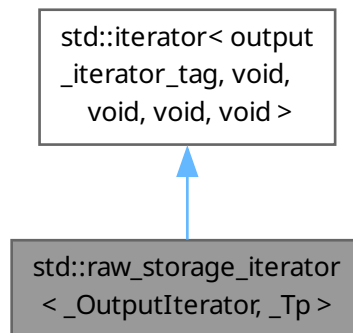
The documentation for this struct was generated from the following file:

- [ratio](#)

#### 5.831 `std::raw_storage_iterator<_OutputIterator, _Tp>` Class Template Reference

```
#include <stl_raw_storage_iter.h>
```

Inheritance diagram for std::raw\_storage\_iterator< \_OutputIterator, \_Tp >:



### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- **raw\_storage\_iterator** ([\\_OutputIterator](#) \_\_x)
- [\\_OutputIterator](#) **base** () const
- [raw\\_storage\\_iterator](#) & **operator\*** ()
- [raw\\_storage\\_iterator](#) & **operator++** ()
- [raw\\_storage\\_iterator](#) **operator++** (int)
- [raw\\_storage\\_iterator](#) & **operator=** ([\\_Tp](#) &&\_\_element)
- [raw\\_storage\\_iterator](#) & **operator=** (const [\\_Tp](#) &&\_\_element)

### Protected Attributes

- [\\_OutputIterator](#) **\_M\_iter**

#### 5.831.1 Detailed Description

```
template<class _OutputIterator, class _Tp>
class std::raw_storage_iterator< _OutputIterator, _Tp >
```

This iterator class lets algorithms store their results into uninitialized memory.

#### 5.831.2 Member Typedef Documentation

##### difference\_type

typedef void [std::iterator](#)< [output\\_iterator\\_tag](#), void, void, void, void >::difference\_type [inherited]  
 Distance between iterators is represented as this type.

### iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category [inherited]
```

One of the [tag types](#).

### pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

### reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value\_type.

### value\_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
```

The type "pointed to" by the iterator.

The documentation for this class was generated from the following file:

- [stl\\_raw\\_storage\\_iter.h](#)

## 5.832 \_\_gnu\_cxx::rb\_tree< \_Key, \_Value, \_KeyOfValue, \_Compare, \_Alloc > Struct Template Reference

```
#include <rb_tree>
```

### Public Types

- `template<typename _Iter>`  
  using **\_\_same\_value\_type**
- `typedef std::Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > _Base`
- `typedef _Base::allocator_type allocator_type`
- `typedef _Node_traits::_Const_iterator const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse\_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Node_traits::_Iterator iterator`
- `typedef _Key key_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef std::reverse\_iterator< iterator > reverse_iterator`
- `typedef size_t size_type`
- `typedef _Val value_type`

### Public Member Functions

- **rb\_tree** (const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11")) iterator erase(const\_iterator \_\_first
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11")) iterator erase(const\_iterator \_\_position)
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11")) iterator erase(iterator \_\_position)

- `bool __rb_verify () const`
- `template<typename _Iterator>`  
`void _M_assign_equal (_Iterator, _Iterator)`
- `template<typename _Iterator>`  
`void _M_assign_unique (_Iterator, _Iterator)`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`size_type _M_count_tr (const _Kt &__k) const`
- `template<typename... _Args>`  
`iterator _M_emplace_equal (_Args &&... __args)`
- `template<typename... _Args>`  
`auto _M_emplace_equal (_Args &&... __args) -> iterator`
- `template<typename... _Args>`  
`iterator _M_emplace_hint_equal (const_iterator __pos, _Args &&... __args)`
- `template<typename... _Args>`  
`auto _M_emplace_hint_equal (const_iterator __pos, _Args &&... __args) -> iterator`
- `template<typename... _Args>`  
`iterator _M_emplace_hint_unique (const_iterator __pos, _Args &&... __args)`
- `template<typename... _Args>`  
`auto _M_emplace_hint_unique (const_iterator __pos, _Args &&... __args) -> iterator`
- `template<typename... _Args>`  
`pair< iterator, bool > _M_emplace_unique (_Args &&... __args)`
- `template<typename... _Args>`  
`auto _M_emplace_unique (_Args &&... __args) -> pair< iterator, bool >`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`pair< iterator, iterator > _M_equal_range_tr (const _Kt &__k)`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`pair< const_iterator, const_iterator > _M_equal_range_tr (const _Kt &__k) const`
- `size_type _M_erase_unique (const key_type &__x)`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`iterator _M_find_tr (const _Kt &__k)`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`const_iterator _M_find_tr (const _Kt &__k) const`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_equal_pos (const key_type &__k)`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_hint_equal_pos (const_iterator __pos, const key_type &__k)`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_hint_unique_pos (const_iterator __pos, const key_type &__k)`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_unique_pos (const key_type &__k)`
- `const _Node_allocator & _M_get_Node_allocator () const noexcept`
- `_Node_allocator & _M_get_Node_allocator () noexcept`
- `template<typename _Arg>`  
`iterator _M_insert_equal (_Arg &&__x)`
- `template<typename _Arg>`  
`iterator _M_insert_equal_ (const_iterator __pos, _Arg &&__x)`
- `template<typename _Arg, typename _NodeGen>`  
`iterator _M_insert_equal_ (const_iterator __pos, _Arg &&__x, _NodeGen &)`
- `template<typename _InputIterator>`  
`__enable_if_t< __same_value_type< _InputIterator >::value > _M_insert_range_equal (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`__enable_if_t< !__same_value_type< _InputIterator >::value > _M_insert_range_equal (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`__enable_if_t< __same_value_type< _InputIterator >::value > _M_insert_range_unique (_InputIterator __first, _InputIterator __last)`



- `template<typename _InputIterator>`  
`__enable_if_t<!\_\_same\_value\_type< _InputIterator >::value > _M_insert_range_unique (_InputIterator __↵`  
`first, _InputIterator __last)`
- `template<typename _Arg>`  
`pair< typename _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >::iterator, bool > _M_insert_unique`  
`( _Arg &&__v)`
- `template<typename _Arg>`  
`pair< iterator, bool > _M_insert_unique ( _Arg &&__x)`
- `template<typename _Arg>`  
`iterator _M_insert_unique (const_iterator __pos, _Arg &&__x)`
- `template<typename _Arg, typename _NodeGen>`  
`iterator _M_insert_unique (const_iterator __pos, _Arg &&__x, _NodeGen &)`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`_Base_ptr _M_lower_bound_tr (const _Kt &__k) const`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`_Base_ptr _M_upper_bound_tr (const _Kt &__k) const`
- `const_iterator begin ()` const noexcept
- `iterator begin ()` noexcept
- `void clear ()` noexcept
- `size_type count (const key_type &__k) const`
- `bool empty ()` const noexcept
- `const_iterator end ()` const noexcept
- `iterator end ()` noexcept
- `pair< iterator, iterator > equal_range (const key_type &__k)`
- `pair< const_iterator, const_iterator > equal_range (const key_type &__k) const`
- `size_type erase (const key_type &__x)`
- `iterator find (const key_type &__k)`
- `const_iterator find (const key_type &__k) const`
- `allocator_type get_allocator ()` const noexcept
- `return iterator (__last._M_node)`
- `_Compare key_comp ()` const
- `iterator lower_bound (const key_type &__k)`
- `const_iterator lower_bound (const key_type &__k) const`
- `size_type max_size ()` const noexcept
- `const\_reverse\_iterator rbegin ()` const noexcept
- `reverse\_iterator rbegin ()` noexcept
- `const\_reverse\_iterator rend ()` const noexcept
- `reverse\_iterator rend ()` noexcept
- `size_type size ()` const noexcept
- `void swap (_Rb_tree &__t)` noexcept(*/\*conditional \*/*)
- `iterator upper_bound (const key_type &__k)`
- `const_iterator upper_bound (const key_type &__k) const`

## Public Attributes

- `const_iterator __last`

## Protected Types

- `typedef _Node_traits::_Base_ptr _Base_ptr`
- `typedef _Node_traits::_Header_t _Header_t`
- `typedef _Node_traits::_Node_ptr _Node_ptr`

### Protected Member Functions

- `_Base_ptr _M_begin ()` const noexcept
- `_Node_ptr _M_begin_node ()` const noexcept
- `template<bool _MoveValue, typename _NodeGen>`  
`_Node_ptr _M_clone_node (_Node_ptr __x, _NodeGen &__node_gen)`
- `template<typename... _Args>`  
`void _M_construct_node (_Node_ptr __node, _Args &&... __args)`
- `template<typename... _Args>`  
`_Node_ptr _M_create_node (_Args &&... __args)`
- `void _M_destroy_node (_Node_ptr __p)` noexcept
- `void _M_drop_node (_Node_ptr __p)` noexcept
- `_Base_ptr _M_end ()` const noexcept
- `_Node_ptr _M_get_node ()`
- `template<typename _Key1, typename _Key2>`  
`bool _M_key_compare (const _Key1 &__k1, const _Key2 &__k2) const`
- `_Base_ptr _M_leftmost ()` const noexcept
- `_Base_ptr & _M_leftmost ()` noexcept
- `void _M_put_node (_Node_ptr __p)` noexcept
- `_Base_ptr _M_rightmost ()` const noexcept
- `_Base_ptr & _M_rightmost ()` noexcept
- `_Base_ptr _M_root ()` const noexcept
- `_Base_ptr & _M_root ()` noexcept

### Static Protected Member Functions

- `static const _Key & _S_key (_Base_ptr __x)`
- `static const _Key & _S_key (_Node_ptr __x)`
- `static const _Key & _S_key (const _Node &__node)`
- `static _Base_ptr _S_left (_Base_ptr __x)` noexcept
- `static _Node_ptr _S_left (_Node_ptr __x)`
- `static _Base_ptr _S_right (_Base_ptr __x)` noexcept
- `static _Node_ptr _S_right (_Node_ptr __x)` noexcept

### Protected Attributes

- `_Rb_tree_impl<_Compare > _M_impl`

#### 5.832.1 Detailed Description

`template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = std::allocator<_↵`  
`Value>>`

`struct __gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc >`

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_↵_style.html)

The documentation for this struct was generated from the following file:

- [rb\\_tree](#)

### 5.833 `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

```
#include <rb_tree_.hpp>
```

#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `base_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `rb_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key\_const\_pointer**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_pointer` **key\_pointer**
- typedef `base_type::key_reference` **key\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `base_type::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `base_type::mapped_const_reference` **mapped\_const\_reference**
- typedef `base_type::mapped_pointer` **mapped\_pointer**
- typedef `base_type::mapped_reference` **mapped\_reference**
- typedef `base_type::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**
- typedef `base_type::node_update` **node\_update**
- typedef `base_type::const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `base_type::reverse_iterator` **reverse\_iterator**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `base_type::value_type` **value\_type**

#### Public Member Functions

- **rb\_tree\_map** (const `Cmp_Fn` &)
- **rb\_tree\_map** (const `Cmp_Fn` &, const `node_update` &)
- **rb\_tree\_map** (const `rb_tree_map`< `Key`, `Mapped`, `Cmp_Fn`, `Node_And_It_Traits`, `_Alloc` > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename `It`>  
void **copy\_from\_range** (`It`, `It`)
- bool **empty** () const

- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator)
- bool **erase** (key\_const\_reference)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- `std::pair< point_iterator, bool >` **insert** (const\_reference)
- void **join** (`rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_begin** () const
- node\_iterator **node\_end** ()
- node\_const\_iterator **node\_end** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, `rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- void **swap** (`bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- void **swap** (`rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Types

- typedef node\_alloc\_traits::value\_type **node**
- typedef node\_alloc\_traits::allocator\_type **node\_allocator**
- typedef traits\_type::null\_node\_update\_pointer **null\_node\_update\_pointer**
- typedef `types_traits< Key, Mapped, _Alloc, false >` **traits\_base**

## Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- template<typename Node\_Update\_>  
void **apply\_update** (node\_pointer, Node\_Update\_\*)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- **std::pair**< node\_pointer, bool > **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- **std::pair**< point\_iterator, bool > **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_subtree\_size** (node\_pointer)
- template<typename Node\_Update\_>  
void **update\_to\_top** (node\_pointer, Node\_Update\_\*)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

## Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

## Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

## Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

### 5.833.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Red-Black tree.

This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

### 5.833.2 Member Function Documentation

#### `node_begin()` [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
```

```
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin
() [inline], [inherited]
```

Returns a `node_iterator` corresponding to the node at the root of the tree.

#### `node_begin()` [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
```

```
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node↵
begin () const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to the node at the root of the tree.

#### `node_end()` [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
```

```
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()
[inline], [inherited]
```

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

#### `node_end()` [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
```

```
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node↵
end () const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- [rb\\_tree\\_.hpp](#)

## 5.834 `__gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >` Struct Template Reference

```
#include <node.hpp>
```

### Public Types

- typedef [rebind\\_traits](#)< `_Alloc`, `metadata_type` >::const\_reference **metadata\_const\_reference**
- typedef [rebind\\_traits](#)< `_Alloc`, `metadata_type` >::reference **metadata\_reference**
- typedef `Metadata` **metadata\_type**
- typedef [rebind\\_traits](#)< `_Alloc`, [rb\\_tree\\_node\\_](#) >::pointer **node\_pointer**
- typedef [rebind\\_traits](#)< `_Alloc`, [rb\\_tree\\_node\\_](#) >::size\_type **size\_type**
- typedef `Value_Type` **value\_type**

### Public Member Functions

- metadata\_reference **get\_metadata** ()
- metadata\_const\_reference **get\_metadata** () const
- bool **special** () const

### Public Attributes

- metadata\_type **m\_metadata**
- node\_pointer **m\_p\_left**
- node\_pointer **m\_p\_parent**
- node\_pointer **m\_p\_right**
- bool **m\_red**
- size\_type **m\_subtree\_size**
- value\_type **m\_value**

#### 5.834.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >
```

Node for Red-Black trees.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/node.hpp](#)

#### 5.835 \_\_gnu\_pbds::rb\_tree\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::rb\_tree\_tag:



### 5.835.1 Detailed Description

Red-black tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.836 \_\_gnu\_pbds::detail::rc< \_Node, \_Alloc > Class Template Reference

```
#include <rc.hpp>
```

### Public Types

- typedef entry\_const\_pointer **const\_iterator**
- typedef node\_pointer **entry**

### Public Member Functions

- **rc** (const [rc](#) &)
- const const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- const const\_iterator **end** () const



- void **pop** ()
- void **push** (entry)
- size\_type **size** () const
- void **swap** (rc &)
- node\_pointer **top** () const

### 5.836.1 Detailed Description

```
template<typename _Node, typename _Alloc>
class __gnu_pbds::detail::rc< _Node, _Alloc >
```

Redundant binary counter.

The documentation for this class was generated from the following file:

- [rc.hpp](#)

## 5.837 \_\_gnu\_pbds::detail::rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

```
#include <rc_binomial_heap.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc >:



### Public Types

- typedef base\_type::allocator\_type **allocator\_type**
- typedef base\_type::cmp\_fn **cmp\_fn**
- typedef [base\\_type::const\\_iterator](#) **const\_iterator**
- typedef base\_type::const\_pointer **const\_pointer**
- typedef base\_type::const\_reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef [base\\_type::iterator](#) **iterator**
- typedef [base\\_type::point\\_const\\_iterator](#) **point\_const\_iterator**
- typedef [base\\_type::point\\_iterator](#) **point\_iterator**
- typedef base\_type::pointer **pointer**
- typedef base\_type::reference **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef Value\_Type **value\_type**

### Public Member Functions

- **rc\_binomial\_heap** (const Cmp\_Fn &)
- **rc\_binomial\_heap** (const [rc\\_binomial\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- [iterator](#) **begin** ()
- [const\\_iterator](#) **begin** () const

- void **clear** ()
- bool **empty** () const
- **iterator end** ()
- **const\_iterator end** () const
- void **erase** (**point\_iterator**)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **join** (**binomial\_heap\_base**< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **join** (**rc\_binomial\_heap**< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type **max\_size** () const
- void **modify** (**point\_iterator**, const\_reference)
- void **pop** ()
- **point\_iterator push** (const\_reference)
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, **binomial\_heap\_base**< Value\_Type, Cmp\_Fn, \_Alloc > &)
- template<typename Pred>  
void **split** (Pred, **rc\_binomial\_heap**< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** (**left\_child\_next\_sibling\_heap**< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- void **swap** (**rc\_binomial\_heap**< Value\_Type, Cmp\_Fn, \_Alloc > &)
- const\_reference **top** () const

### Protected Types

- typedef **base\_type::node** **node**
- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef **std::pair**< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It>  
void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** (**binomial\_heap\_base**< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** (**left\_child\_next\_sibling\_heap** &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 5.837.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Redundant-counter binomial heap.

The documentation for this class was generated from the following file:

- [rc\\_binomial\\_heap.hpp](#)

### 5.838 \_\_gnu\_pbds::rc\_binomial\_heap\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::rc\_binomial\_heap\_tag:



#### 5.838.1 Detailed Description

Redundant-counter binomial-heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.839 \_\_gnu\_pbds::detail::rebind\_traits< \_Alloc, T > Struct Template Reference

```
#include <types_traits.hpp>
```

## Public Types

- using **const\_reference**
- using **reference**

### 5.839.1 Detailed Description

```
template<typename _Alloc, typename T>
struct __gnu_pbds::detail::rebind_traits<_Alloc, T >
```

Consistent API for accessing allocator-related types.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 5.840 std::filesystem::recursive\_directory\_iterator Class Reference

```
#include <filesystem>
```

## Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [input\\_iterator\\_tag](#) **iterator\_category**
- typedef const [directory\\_entry](#) \* **pointer**
- typedef const [directory\\_entry](#) & **reference**
- typedef [directory\\_entry](#) **value\_type**

## Public Member Functions

- **recursive\_directory\_iterator** (const [path](#) &\_\_p)
- **recursive\_directory\_iterator** (const [path](#) &\_\_p, [directory\\_options](#) \_\_options)
- **recursive\_directory\_iterator** (const [path](#) &\_\_p, [directory\\_options](#) \_\_options, [error\\_code](#) &\_\_ec)
- **recursive\_directory\_iterator** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- **recursive\_directory\_iterator** (const [recursive\\_directory\\_iterator](#) &)=default
- **recursive\_directory\_iterator** ([recursive\\_directory\\_iterator](#) &&)=default
- int **depth** () const noexcept
- void **disable\_recursion\_pending** () noexcept
- [recursive\\_directory\\_iterator](#) & **increment** ([error\\_code](#) &\_\_ec)
- const [directory\\_entry](#) & **operator\*** () const noexcept
- [recursive\\_directory\\_iterator](#) & **operator++** ()
- [\\_\\_directory\\_iterator\\_proxy](#) **operator++** (int)
- const [directory\\_entry](#) \* **operator->** () const noexcept
- [recursive\\_directory\\_iterator](#) & **operator=** (const [recursive\\_directory\\_iterator](#) &\_\_rhs) noexcept
- [recursive\\_directory\\_iterator](#) & **operator=** ([recursive\\_directory\\_iterator](#) &&\_\_rhs) noexcept
- bool **operator==** ([default\\_sentinel\\_t](#)) const noexcept
- [directory\\_options](#) **options** () const noexcept
- void **pop** ()
- void **pop** ([error\\_code](#) &)
- bool **recursion\_pending** () const noexcept

## Friends

- uintmax\_t **filesystem::remove\_all** (const [path](#) &)
- uintmax\_t **filesystem::remove\_all** (const [path](#) &, [error\\_code](#) &)
- bool **operator==** (const [recursive\\_directory\\_iterator](#) &\_\_lhs, const [recursive\\_directory\\_iterator](#) &\_\_rhs) noexcept

### 5.840.1 Detailed Description

Iterator type for recursively traversing a directory hierarchy.

Since

C++17

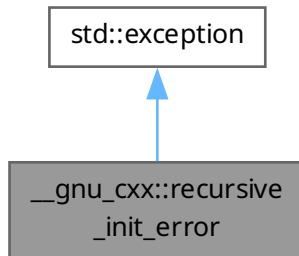
The documentation for this class was generated from the following file:

- [bits/fs\\_dir.h](#)

### 5.841 \_\_gnu\_cxx::recursive\_init\_error Class Reference

```
#include <cxxabi.h>
```

Inheritance diagram for \_\_gnu\_cxx::recursive\_init\_error:



#### Public Member Functions

- virtual const char \* [what](#) () const noexcept

### 5.841.1 Detailed Description

Exception thrown by `__cxa_guard_acquire`.

C++ 2011 6.7 [stmt.dcl]/4: If control re-enters the declaration recursively while the variable is being initialized, the behavior is undefined.

Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

### 5.841.2 Member Function Documentation

#### what()

```
virtual const char * std::exception::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad\\_alloc](#), [std::bad\\_cast](#), [std::bad\\_exception](#), [std::bad\\_function\\_call](#), [std::bad\\_typeid](#), [std::bad\\_weak\\_ptr](#), [std::experimental::filesystem::v1::filesystem\\_error](#), [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#), [std::filesystem::filesystem\\_error](#), [std::future\\_error](#), [std::logic\\_error](#), and [std::runtime\\_error](#).

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

## 5.842 std::recursive\_mutex Class Reference

```
#include <mutex>
```

### Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

### Public Member Functions

- **recursive\_mutex** (const [recursive\\_mutex](#) &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** () noexcept
- [recursive\\_mutex](#) & **operator=** (const [recursive\\_mutex](#) &)=delete
- bool **try\_lock** () noexcept
- void **unlock** ()

#### 5.842.1 Detailed Description

The standard recursive mutex type.

A recursive mutex can be locked more than once by the same thread. Other threads cannot lock the mutex until the owning thread unlocks it as many times as it was locked.

Since

C++11

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.843 std::recursive\_timed\_mutex Class Reference

```
#include <mutex>
```

### Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

### Public Member Functions

- **recursive\_timed\_mutex** (const [recursive\\_timed\\_mutex](#) &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** () noexcept
- [recursive\\_timed\\_mutex](#) & **operator=** (const [recursive\\_timed\\_mutex](#) &)=delete
- bool **try\_lock** () noexcept
- template<class \_Rep, class \_Period>  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<class \_Clock, class \_Duration>  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

### Friends

- class **\_\_timed\_mutex\_impl**< [recursive\\_timed\\_mutex](#) >

### 5.843.1 Detailed Description

The standard recursive timed mutex type.

A recursive mutex that supports a timeout when trying to acquire the lock. A recursive mutex can be locked more than once by the same thread. Other threads cannot lock the mutex until the owning thread unlocks it as many times as it was locked.

Since

C++11

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.844 std::bitset<\_Nb>::reference Class Reference

```
#include <bitset>
```

### Public Member Functions

- **reference** (const [reference](#) &)=default
- constexpr [reference](#) & **flip** () noexcept
- constexpr **operator bool** () const noexcept
- constexpr [reference](#) & **operator=** (bool \_\_x) noexcept
- constexpr [reference](#) & **operator=** (const [reference](#) &\_\_j) noexcept
- constexpr bool **operator~** () const noexcept

### Friends

- class **bitset**

### 5.844.1 Detailed Description

```
template<size_t _Nb>
```

```
class std::bitset<_Nb>::reference
```

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)

The documentation for this class was generated from the following file:

- [bitset](#)

## 5.845 std::tr2::dynamic\_bitset<\_WordT, \_Alloc>::reference Class Reference

```
#include <dynamic_bitset>
```

### Public Member Functions

- **reference** ([dynamic\\_bitset](#) &\_\_b, size\_type \_\_pos) noexcept
- [reference](#) & **flip** () noexcept
- **operator bool** () const noexcept
- [reference](#) & **operator=** (bool \_\_x) noexcept
- [reference](#) & **operator=** (const [reference](#) &\_\_j) noexcept
- bool **operator~** () const noexcept

## Friends

- class `dynamic_bitset`

### 5.845.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset<_WordT, _Alloc>::reference
```

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from `bool` are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this "bit %reference" is 64 times the size of an actual bit. Ha.)

The documentation for this class was generated from the following file:

- [dynamic\\_bitset](#)

## 5.846 std::reference\_wrapper<\_Tp> Class Template Reference

```
#include <refwrap.h>
```

### Public Types

- typedef `_Tp type`

### Public Member Functions

- `template<typename _Up, typename = __not_same<_Up>, typename = decltype(reference_wrapper::_S_fun(std::declval<_Up>()))> constexpr reference_wrapper(_Up && __uref) noexcept(noexcept(reference_wrapper::_S_fun(std::declval<_Up>())>))`
- `reference_wrapper(const reference_wrapper &)=default`
- `constexpr _Tp & get() const noexcept`
- `constexpr operator _Tp &() const noexcept`
- `template<typename... _Args> constexpr __invoke_result<_Tp &, _Args...>::type operator()(_Args &&... __args) const noexcept(__is_nothrow_invocable<_Tp &, _Args...>::value)`
- `reference_wrapper & operator= (const reference_wrapper &)=default`

### Related Symbols

(Note that these are not member symbols.)

- `template<typename _Tp> constexpr reference_wrapper<_Tp> ref(_Tp & __t) noexcept`
- `template<typename _Tp> constexpr reference_wrapper<const _Tp> cref(const _Tp & __t) noexcept`
- `template<typename _Tp> constexpr reference_wrapper<_Tp> ref(reference_wrapper<_Tp> __t) noexcept`
- `template<typename _Tp> constexpr reference_wrapper<const _Tp> cref(reference_wrapper<_Tp> __t) noexcept`

### 5.846.1 Detailed Description

```
template<typename _Tp>
class std::reference_wrapper<_Tp>
```

Primary class template for `reference_wrapper`.



### 5.846.2 Friends And Related Symbol Documentation

#### **cref()** [1/2]

```
template<typename _Tp>
reference_wrapper< const _Tp > cref (
    const _Tp & __t) [related]
```

Denotes a const reference should be taken to a variable.

#### **cref()** [2/2]

```
template<typename _Tp>
reference_wrapper< const _Tp > cref (
    reference_wrapper< _Tp > __t) [related]
```

std::cref overload to prevent wrapping a reference\_wrapper

#### **ref()** [1/2]

```
template<typename _Tp>
reference_wrapper< _Tp > ref (
    _Tp & __t) [related]
```

Denotes a reference should be taken to a variable.

#### **ref()** [2/2]

```
template<typename _Tp>
reference_wrapper< _Tp > ref (
    reference_wrapper< _Tp > __t) [related]
```

std::ref overload to prevent wrapping a reference\_wrapper

The documentation for this class was generated from the following files:

- [type\\_traits](#)
- [refwrap.h](#)

### 5.847 std::regex\_error Class Reference

```
#include <regex>
```

Inheritance diagram for `std::regex_error`:



### Public Member Functions

- [regex\\_error](#) ([error\\_type](#) \_\_ecode)
- [regex\\_constants::error\\_type code](#) () const noexcept
- virtual const char \* [what](#) () const noexcept

#### 5.847.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Since

C++11

#### 5.847.2 Constructor & Destructor Documentation

##### `regex_error()`

```
std::regex_error::regex_error (
    error\_type __ecode) [explicit]
```

Constructs a `regex_error` object.

Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__ecode</code> | the regex error code. |
|----------------------|-----------------------|

References [regex\\_error\(\)](#).

Referenced by [regex\\_error\(\)](#).

#### 5.847.3 Member Function Documentation

##### `code()`

```
regex\_constants::error\_type std::regex_error::code () const [inline], [noexcept]
```

Gets the regex error code.

Returns

the regex error code.

References [code\(\)](#).

Referenced by [code\(\)](#).

**what()**

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), and [std::filesystem::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [regex\\_error.h](#)

## 5.848 [std::regex\\_iterator<\\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits>](#) Class Template Reference

```
#include <regex>
```

### Public Types

- typedef [std::ptrdiff\\_t](#) **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef [std::input\\_iterator\\_tag](#) **iterator\_concept**
- typedef const value\_type \* **pointer**
- typedef const value\_type & **reference**
- typedef [basic\\_regex<\\_Ch\\_type, \\_Rx\\_traits>](#) **regex\_type**
- typedef [match\\_results<\\_Bi\\_iter>](#) **value\_type**

### Public Member Functions

- [regex\\_iterator](#) ()=default
- [regex\\_iterator](#) (\_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, [regex\\_constants::match\\_flag\\_type](#) \_\_m= [regex\\_constants::match\\_default](#))
- [regex\\_iterator](#) (\_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))=delete
- [regex\\_iterator](#) (const [regex\\_iterator](#) &)=default
- const value\_type & [operator\\*](#) () const noexcept
- [regex\\_iterator](#) & [operator++](#) ()
- [regex\\_iterator](#) [operator++](#) (int)
- const value\_type \* [operator->](#) () const noexcept
- [regex\\_iterator](#) & [operator=](#) (const [regex\\_iterator](#) &)=default
- bool [operator==](#) (const [regex\\_iterator](#) &) const noexcept
- bool [operator==](#) ([default\\_sentinel\\_t](#)) const noexcept

### 5.848.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>>
```

```
class std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>
```

An iterator adaptor that will provide repeated calls of [regex\\_search](#) over a range until no more matches remain.

Since

C++11

### 5.848.2 Constructor & Destructor Documentation

#### `regex_iterator()` [1/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator () [default]
```

Provides a singular iterator, useful for indicating one-past-the-end of a range.

Referenced by `std::regex_iterator<const char*>::regex_iterator()`, and `operator==()`.

#### `regex_iterator()` [2/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator (
    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a `regex_iterator`...

#### Parameters

|                   |                                                    |
|-------------------|----------------------------------------------------|
| <code>__a</code>  | [IN] The start of a text range to search.          |
| <code>__b</code>  | [IN] One-past-the-end of the text range to search. |
| <code>__re</code> | [IN] The regular expression to match.              |
| <code>__m</code>  | [IN] Policy flags for match rules.                 |

#### `regex_iterator()` [3/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator (
    const regex_iterator<_Bi_iter, _Ch_type, _Rx_traits> & ) [default]
```

Copy constructs a `regex_iterator`.

### 5.848.3 Member Function Documentation

#### `operator*()`

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type & std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator* () const
[inline], [noexcept]
```

Dereferences a `regex_iterator`.

**operator++()** [1/2]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ()
```

Increments a `regex_iterator`.

References `std::regex_constants::match_continuous`, `std::regex_constants::match_not_null`, `std::regex_constants::match_prev_avail`, and `std::regex_search()`.

**operator++()** [2/2]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
    int ) [inline]
```

Postincrements a `regex_iterator`.

**operator->()**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type * std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> () const
[inline], [noexcept]
```

Selects a `regex_iterator` member.

**operator=()**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator & std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
    const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & ) [default]
```

Copy assigns one `regex_iterator` to another.

**operator==()**

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
    const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const [noexcept]
```

Tests the equivalence of two `regex` iterators.

References `regex_iterator()`.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

## 5.849 `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

```
#include <regex>
```

**Public Types**

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `std::input_iterator_tag` **iterator\_concept**

- typedef const [value\\_type](#) \* **pointer**
- typedef const [value\\_type](#) & **reference**
- typedef [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > **regex\_type**
- typedef [sub\\_match](#)< \_Bi\_iter > **value\_type**

## Public Member Functions

- [regex\\_token\\_iterator](#) ()
- template<std::size\_t \_Nm>  
[regex\\_token\\_iterator](#) (\_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, const int(&\_\_submatches)[\_Nm], [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- [regex\\_token\\_iterator](#) (\_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, const [std::vector](#)< int > &\_\_submatches, [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- [regex\\_token\\_iterator](#) (\_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, [initializer\\_list](#)< int > \_\_submatches, [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- [regex\\_token\\_iterator](#) (\_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, int \_\_submatch=0, [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- template<std::size\_t \_Nm>  
**regex\_token\_iterator** (\_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, const int(&)[\_Nm], [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))
- **regex\_token\_iterator** (\_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, const [std::vector](#)< int > &, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))
- **regex\_token\_iterator** (\_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, [initializer\\_list](#)< int >, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))
- **regex\_token\_iterator** (\_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, int=0, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))
- [regex\\_token\\_iterator](#) (const [regex\\_token\\_iterator](#) &\_\_rhs)
- const [value\\_type](#) & **operator\*** () const
- [regex\\_token\\_iterator](#) & **operator++** ()
- [regex\\_token\\_iterator](#) **operator++** (int)
- const [value\\_type](#) \* **operator->** () const
- [regex\\_token\\_iterator](#) & **operator=** (const [regex\\_token\\_iterator](#) &\_\_rhs)
- bool **operator==** (const [regex\\_token\\_iterator](#) &\_\_rhs) const
- bool **operator==** ([default\\_sentinel\\_t](#)) const noexcept

### 5.849.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>>
class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a `std::sub_match` object.

Since

C++11

### 5.849.2 Constructor & Destructor Documentation

**regex\_token\_iterator**() [1/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::regex\_token\_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator () [inline]
```

Default constructs a `regex_token_iterator`.

A default-constructed `regex_token_iterator` is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

Referenced by [operator++\(\)](#), [operator=\(\)](#), and [operator==\(\)](#).

**regex\_token\_iterator()** [2/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    int __submatch = 0,
    regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a regex\_token\_iterator...

**Parameters**

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__a</code>        | [IN] The start of the text to search.                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>__b</code>        | [IN] One-past-the-end of the text to search.                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>__re</code>       | [IN] The regular expression to search for.                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>__submatch</code> | [IN] Which submatch to return. There are some special values for this parameter: <ul style="list-style-type: none"> <li>• -1 each enumerated subexpression does NOT match the regular expression (aka field splitting)</li> <li>• 0 the entire string matching the subexpression is returned for each match within the text.</li> <li>• &gt;0 enumerates only the indicated subexpression from a match within the text.</li> </ul> |
| <code>__m</code>        | [IN] Policy flags for match rules.                                                                                                                                                                                                                                                                                                                                                                                                 |

**regex\_token\_iterator()** [3/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    const std::vector< int > & __submatches,
    regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a regex\_token\_iterator...

**Parameters**

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

**regex\_token\_iterator()** [4/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
```

```

    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    initializer_list< int > __submatches,
    regex_constants::match_flag_type __m = regex_constants::match_default) [inline]

```

Constructs a regex\_token\_iterator...

#### Parameters

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

#### regex\_token\_iterator() [5/6]

```

template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
template<std::size_t _Nm>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    const int (&) __submatches[_Nm],
    regex_constants::match_flag_type __m = regex_constants::match_default) [inline]

```

Constructs a regex\_token\_iterator...

#### Parameters

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

#### regex\_token\_iterator() [6/6]

```

template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
    const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) [inline]

```

Copy constructs a regex\_token\_iterator.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__rhs</code> | [IN] A regex_token_iterator to copy. |
|--------------------|--------------------------------------|



### 5.849.3 Member Function Documentation

#### **operator\*()**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ()
const [inline]
```

Dereferences a `regex_token_iterator`.

#### **operator++()** [1/2]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ()
Ch_type, _Rx_traits >::operator++ ()
```

Increments a `regex_token_iterator`.

References [regex\\_token\\_iterator\(\)](#).

#### **operator++()** [2/2]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_token_iterator std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
int ) [inline]
```

Postincrements a `regex_token_iterator`.

#### **operator->()**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type * std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ()
const [inline]
```

Selects a `regex_token_iterator` member.

#### **operator=()**

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs)
```

Assigns a `regex_token_iterator` to another.

#### Parameters

|                    |                                                   |
|--------------------|---------------------------------------------------|
| <code>__rhs</code> | [IN] A <code>regex_token_iterator</code> to copy. |
|--------------------|---------------------------------------------------|

References [regex\\_token\\_iterator\(\)](#).

#### **operator==()**

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const
```

Compares a `regex_token_iterator` to another for equality.

References [regex\\_token\\_iterator\(\)](#).

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

## 5.850 `std::regex_traits<_Ch_type>` Class Template Reference

```
#include <regex>
```

### Public Types

- typedef `_RegexMask` **char\_class\_type**
- typedef `_Ch_type` **char\_type**
- typedef [std::locale](#) **locale\_type**
- typedef [std::basic\\_string](#)< `char_type` > **string\_type**

### Public Member Functions

- [regex\\_traits](#) ()
- [locale\\_type](#) `getloc` () const
- [locale\\_type](#) `imbue` ([locale\\_type](#) \_\_loc)
- bool [isctype](#) (`_Ch_type` \_\_c, `char_class_type` \_\_f) const
- template<typename `_Fwd_iter`>  
`char_class_type` [lookup\\_classname](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last, bool \_\_icase=false) const
- template<typename `_Fwd_iter`>  
[string\\_type](#) [lookup\\_collatename](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const
- template<typename `_Fwd_iter`>  
[string\\_type](#) [transform](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const
- template<typename `_Fwd_iter`>  
[string\\_type](#) [transform\\_primary](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const
- `char_type` [translate](#) (`char_type` \_\_c) const
- `char_type` [translate\\_nocase](#) (`char_type` \_\_c) const
- int [value](#) (`_Ch_type` \_\_ch, int \_\_radix) const

### Static Public Member Functions

- static `std::size_t` [length](#) (const `char_type` \*\_\_p)

### Protected Attributes

- [locale\\_type](#) `_M_locale`

#### 5.850.1 Detailed Description

```
template<typename _Ch_type>
class std::regex_traits<_Ch_type>
```

Describes aspects of a regular expression.

A regular expression traits class that satisfies the requirements of section [28.7].

The class `regex` is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Since

C++11

### 5.850.2 Constructor & Destructor Documentation

#### regex\_traits()

```
template<typename _Ch_type>
std::regex_traits< _Ch_type >::regex_traits () [inline]
Constructs a default traits object.
```

### 5.850.3 Member Function Documentation

#### getloc()

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::getloc () const [inline]
Gets a copy of the current locale in use by the regex_traits object.
Referenced by std::regex\_traits< \_CharT >::lookup\_classname\(\).
```

#### imbue()

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::imbue (
    locale_type __loc) [inline]
Imbues the regex_traits object with a copy of a new locale.
```

##### Parameters

|                    |           |
|--------------------|-----------|
| <code>__loc</code> | A locale. |
|--------------------|-----------|

##### Returns

a copy of the previous locale in use by the regex\_traits object.

##### Note

Calling imbue with a different locale than the one currently in use invalidates all cached data held by \*this.

#### istype()

```
template<typename _Ch_type>
bool std::regex_traits< _Ch_type >::istype (
    _Ch_type __c,
    char_class_type __f) const
Determines if c is a member of an identified class.
```

##### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | a character.                                      |
| <code>__f</code> | a class type (as returned from lookup_classname). |

##### Returns

true if the character `__c` is a member of the classification represented by `__f`, false otherwise.

**Exceptions**

|                            |                                                    |
|----------------------------|----------------------------------------------------|
| <code>std::bad_cast</code> | if the current locale does not have a ctype facet. |
|----------------------------|----------------------------------------------------|

References [std::use\\_facet\(\)](#).

**length()**

```
template<typename _Ch_type>
static std::size_t std::regex\_traits<\_Ch\_type>::length (
    const char_type * __p) [inline], [static]
```

Gives the length of a C-style string starting at `__p`.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__p</code> | a pointer to the start of a character sequence. |
|------------------|-------------------------------------------------|

**Returns**

the number of characters between `*__p` and the first default-initialized value of type `char_type`. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

**lookup\_classname()**

```
template<typename _Ch_type>
template<typename _Fwd_iter>
regex\_traits<\_Ch\_type>::char\_class\_type std::regex\_traits<\_Ch\_type>::lookup\_classname (
    _Fwd_iter __first,
    _Fwd_iter __last,
    bool __icase = false) const
```

Maps one or more characters to a named character classification.

**Parameters**

|                      |                                              |
|----------------------|----------------------------------------------|
| <code>__first</code> | beginning of the character sequence.         |
| <code>__last</code>  | one-past-the-end of the character sequence.  |
| <code>__icase</code> | ignores the case of the classification name. |

**Returns**

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[__first, __last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[[:lower:]]` is the same as `[[:alpha:]]`), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- `d`
- `w`
- `s`

- alnum
- alpha
- blank
- cntrl
- digit
- graph
- lower
- print
- punct
- space
- upper
- xdigit

References [std::use\\_facet\(\)](#).

### lookup\_collatename()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
regex_traits< _Ch_type >::string_type std::regex_traits< _Ch_type >::lookup_collatename (
    _Fwd_iter __first,
    _Fwd_iter __last) const
```

Gets a collation element by name.

#### Parameters

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <code>__first</code> | beginning of the collation element name.        |
| <code>__last</code>  | one-past-the-end of the collation element name. |

#### Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [`__first`, `__last`). Returns an empty string if the character sequence is not a valid collating element.

References [std::use\\_facet\(\)](#).

### transform()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
string_type std::regex_traits< _Ch_type >::transform (
    _Fwd_iter __first,
    _Fwd_iter __last) const [inline]
```

Gets a sort key for a character sequence.

## Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__first</code> | beginning of the character sequence.        |
| <code>__last</code>  | one-past-the-end of the character sequence. |

Returns a sort key for the character sequence designated by the iterator range `[F1, F2)` such that if the character sequence `[G1, G2)` sorts before the character sequence `[H1, H2)` then `v.transform(G1, G2) < v.transform(H1, H2)`. What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

## Returns

a locale-specific sort key equivalent to the input range.

## Exceptions

|                            |                                                      |
|----------------------------|------------------------------------------------------|
| <code>std::bad_cast</code> | if the current locale does not have a collate facet. |
|----------------------------|------------------------------------------------------|

**transform\_primary()**

```
template<typename _Ch_type>
template<typename _Fwd_iter>
string_type std::regex_traits<_Ch_type>::transform_primary (
    _Fwd_iter __first,
    _Fwd_iter __last) const [inline]
```

Gets a sort key for a character sequence, independent of case.

## Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__first</code> | beginning of the character sequence.        |
| <code>__last</code>  | one-past-the-end of the character sequence. |

**Effects:** if `typeid(use_facet<collate<_Ch_type>>(getloc())) == typeid(collate_↔_byname<_Ch_type>)` and the form of the sort key returned by `collate_↔_byname<_Ch_type>::transform(↔__first, __last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

**Todo** Implement this function correctly.

**translate()**

```
template<typename _Ch_type>
char_type std::regex_traits<_Ch_type>::translate (
    char_type __c) const [inline]
```

Performs the identity translation.

## Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>↔_c</code> | A character to the locale-specific character set. |
|------------------|---------------------------------------------------|

## Returns

`↔_c`.

**translate\_nocase()**

```
template<typename _Ch_type>
char_type std::regex_traits< _Ch_type >::translate_nocase (
    char_type __c) const [inline]
```

Translates a character into a case-insensitive equivalent.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | A character to the locale-specific character set. |
|------------------|---------------------------------------------------|

**Returns**

the locale-specific lower-case equivalent of `__c`.

**Exceptions**

|                            |                                                        |
|----------------------------|--------------------------------------------------------|
| <code>std::bad_cast</code> | if the imbued locale does not support the ctype facet. |
|----------------------------|--------------------------------------------------------|

**value()**

```
template<typename _Ch_type>
int std::regex_traits< _Ch_type >::value (
    _Ch_type __ch,
    int __radix) const
```

Converts a digit to an int.

**Parameters**

|                      |                                                                |
|----------------------|----------------------------------------------------------------|
| <code>__ch</code>    | a character representing a digit.                              |
| <code>__radix</code> | the radix if the numeric conversion (limited to 8, 10, or 16). |

**Returns**

the value represented by the digit `__ch` in base `radix` if the character `__ch` is a valid digit in base `radix`; otherwise returns -1.

References [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::hex\(\)](#), and [std::oct\(\)](#).

Referenced by [std::regex\\_traits< \\_CharT >::isctype\(\)](#).

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

**5.851 std::remove\_all\_extents< \_Tp > Struct Template Reference**

```
#include <type_traits>
```

**Public Types**

- using **type**

### 5.851.1 Detailed Description

```
template<typename _Tp>
struct std::remove_all_extents<_Tp>
```

`remove_all_extents`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.852 `std::remove_const<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using `type`

### 5.852.1 Detailed Description

```
template<typename _Tp>
struct std::remove_const<_Tp>
```

`remove_const`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.853 `std::remove_cv<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using `type`

### 5.853.1 Detailed Description

```
template<typename _Tp>
struct std::remove_cv<_Tp>
```

`remove_cv`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.854 `std::remove_extent<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using `type`



#### 5.854.1 Detailed Description

```
template<typename _Tp>
struct std::remove_extent< _Tp >
```

remove\_extent

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.855 std::remove\_pointer< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### Public Types

- using **type**

#### 5.855.1 Detailed Description

```
template<typename _Tp>
struct std::remove_pointer< _Tp >
```

remove\_pointer

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.856 std::remove\_reference< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### Public Types

- using **type**

#### 5.856.1 Detailed Description

```
template<typename _Tp>
struct std::remove_reference< _Tp >
```

remove\_reference

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.857 std::remove\_volatile< \_Tp > Struct Template Reference

```
#include <type_traits>
```

##### Public Types

- using **type**

### 5.857.1 Detailed Description

```
template<typename _Tp>  
struct std::remove_volatile<_Tp >
```

remove\_volatile

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.858 \_\_gnu\_pbds::resize\_error Struct Reference

```
#include <exception.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::resize\_error:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

### 5.858.1 Detailed Description

A container cannot be resized.

### 5.858.2 Member Function Documentation

#### what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

## 5.859 `__gnu_pbds::detail::resize_policy<_Tp>` Class Template Reference

```
#include <resize_policy.hpp>
```

### Public Types

- typedef `_Tp` `size_type`

### Public Member Functions

- `resize_policy` (const [resize\\_policy](#) &other)
- `size_type` `get_new_size_for_arbitrary` (`size_type`) const
- `size_type` `get_new_size_for_grow` () const
- `size_type` `get_new_size_for_shrink` () const
- `bool` `grow_needed` (`size_type`) const
- `void` `notify_arbitrary` (`size_type`)
- `void` `notify_grow_resize` ()
- `void` `notify_shrink_resize` ()
- `bool` `resize_needed_for_grow` (`size_type`) const
- `bool` `resize_needed_for_shrink` (`size_type`) const
- `bool` `shrink_needed` (`size_type`) const
- `void` `swap` ([resize\\_policy](#)<`_Tp`> &)

### Static Public Attributes

- static const `_Tp` `min_size`

#### 5.859.1 Detailed Description

```
template<typename _Tp>  
class __gnu_pbds::detail::resize_policy<_Tp>
```

Resize policy for binary heap.

The documentation for this class was generated from the following file:

- [resize\\_policy.hpp](#)

## 5.860 `std::result_of<_Signature>` Struct Template Reference

#### 5.860.1 Detailed Description

```
template<typename _Signature>  
struct std::result_of<_Signature>
```

`result_of`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.861 std::reverse\_iterator<\_Iterator> Class Template Reference

```
#include <stl_iterator.h>
```

Inheritance diagram for std::reverse\_iterator<\_Iterator>:



### Public Types

- using **difference\_type**
- using **iterator\_category**
- using **iterator\_concept**
- typedef **\_Iterator** **iterator\_type**
- typedef **\_\_traits\_type::pointer** **pointer**
- using **reference**
- using **value\_type**

### Public Member Functions

- constexpr **reverse\_iterator** () noexcept(*/\*conditional \*/*)
- constexpr **reverse\_iterator** (const **reverse\_iterator** &\_\_x) noexcept(*/\*conditional \*/*)
- template<typename **\_Iter**>  
requires **\_\_convertible<\_Iter>**  
constexpr **reverse\_iterator** (const **reverse\_iterator**< **\_Iter** > &\_\_x) noexcept(*/\*conditional \*/*)
- constexpr **reverse\_iterator** (iterator\_type \_\_x) noexcept(*/\*conditional \*/*)
- constexpr iterator\_type **base** () const noexcept(*/\*conditional \*/*)
- constexpr reference **operator\*** () const
- constexpr **reverse\_iterator** **operator+** (difference\_type \_\_n) const
- constexpr **reverse\_iterator** & **operator++** ()
- constexpr **reverse\_iterator** **operator++** (int)
- constexpr **reverse\_iterator** & **operator+=** (difference\_type \_\_n)
- constexpr **reverse\_iterator** **operator-** (difference\_type \_\_n) const
- constexpr **reverse\_iterator** & **operator--** ()
- constexpr **reverse\_iterator** **operator--** (int)
- constexpr **reverse\_iterator** & **operator-=** (difference\_type \_\_n)
- constexpr pointer **operator->** () const
- **reverse\_iterator** & **operator=** (const **reverse\_iterator** &)=default
- template<typename **\_Iter**>  
requires **\_\_convertible<\_Iter>** && assignable\_from<\_Iterator&, const \_Iter&>  
constexpr **reverse\_iterator** & **operator=** (const **reverse\_iterator**< **\_Iter** > &\_\_x) noexcept(*/\*conditional \*/*)
- constexpr reference **operator[]** (difference\_type \_\_n) const

## Protected Types

- typedef `iterator_traits<_Iterator>` `__traits_type`

## Protected Attributes

- `_Iterator` `current`

## Friends

- constexpr `iter_value_reference_t<_Iterator>` `iter_move` (const `reverse_iterator` &\_\_i) noexcept(is\_nothrow\_copy\_constructible\_v<\_Iterator> &&noexcept(ranges::iter\_move(--std::declval<\_Iterator> >())))
- template<indirectly\_swappable<\_Iterator> \_Iter2>  
constexpr void `iter_swap` (const `reverse_iterator` &\_\_x, const `reverse_iterator`<\_Iter2> &\_\_y) noexcept(is\_nothrow\_copy\_constructible\_v<\_Iterator> &&is\_nothrow\_copy\_constructible\_v<\_Iter2> &&noexcept(ranges::iter\_swap(--std::declval<\_Iterator> >(), --std::declval<\_Iter2> >())))

### 5.861.1 Detailed Description

template<typename `_Iterator`>

class `std::reverse_iterator<_Iterator>`

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
*(reverse_iterator(i)) == *(i - 1)
```

*This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2*

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

### 5.861.2 Constructor & Destructor Documentation

`reverse_iterator()` [1/4]

```
template<typename _Iterator>
```

```
std::reverse_iterator<_Iterator>::reverse_iterator () [inline], [constexpr], [noexcept]
```

The default constructor value-initializes member `current`. If it is a pointer, that means it is zero-initialized.

`reverse_iterator()` [2/4]

```
template<typename _Iterator>
```

```
std::reverse_iterator<_Iterator>::reverse_iterator (
    iterator_type __x) [inline], [explicit], [constexpr], [noexcept]
```

This iterator will move in the opposite direction that `x` does.

`reverse_iterator()` [3/4]

```
template<typename _Iterator>
```

```
std::reverse_iterator<_Iterator>::reverse_iterator (
    const reverse_iterator<_Iterator> &__x) [inline], [constexpr], [noexcept]
```

The copy constructor is normal.

**reverse\_iterator()** [4/4]

```
template<typename _Iterator>
template<typename _Iter>
requires __convertible<_Iter>
std::reverse_iterator<_Iterator>::reverse_iterator (
    const reverse_iterator<_Iter> & __x) [inline], [constexpr], [noexcept]
```

A `reverse_iterator` across other types can be copied if the underlying iterator can be converted to the type of `current`.

**5.861.3 Member Function Documentation****base()**

```
template<typename _Iterator>
iterator_type std::reverse_iterator<_Iterator>::base () const [inline], [nodiscard], [constexpr],
[noexcept]
```

**Returns**

`current`, the iterator used for underlying work.

Referenced by `std::operator==()`.

**operator\*()**

```
template<typename _Iterator>
reference std::reverse_iterator<_Iterator>::operator* () const [inline], [nodiscard], [constexpr]
```

**Returns**

A reference to the value at `--current`

This requires that `--current` is dereferenceable.

**Warning**

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: <http://gcc.gnu.org/PR51823>

**operator+()**

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator<_Iterator>::operator+ (
    difference_type __n) const [inline], [nodiscard], [constexpr]
```

**Returns**

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

**operator++()** [1/2]

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator<_Iterator>::operator++ () [inline], [constexpr]
```

**Returns**

`*this`

Decrements the underlying iterator.

**operator++()** [2/2]

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator++ (
    int ) [inline], [constexpr]
```

**Returns**

The original value of `*this`

Decrements the underlying iterator.

**operator+=()**

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator+= (
    difference_type __n) [inline], [constexpr]
```

**Returns**

`*this`

Moves the underlying iterator backwards `__n` steps. The underlying iterator must be a Random Access Iterator.

**operator-()**

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator- (
    difference_type __n) const [inline], [nodiscard], [constexpr]
```

**Returns**

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

**operator--()** [1/2]

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator-- () [inline], [constexpr]
```

**Returns**

`*this`

Increments the underlying iterator.

**operator--()** [2/2]

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator-- (
    int ) [inline], [constexpr]
```

**Returns**

A `reverse_iterator` with the previous value of `*this`

Increments the underlying iterator.

**operator-=( )**

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator-= (
    difference_type __n) [inline], [constexpr]
```

**Returns**

\*this

Moves the underlying iterator forwards `__n` steps. The underlying iterator must be a Random Access Iterator.

**operator->()**

```
template<typename _Iterator>
pointer std::reverse_iterator< _Iterator >::operator-> () const [inline], [nodiscard], [constexpr]
```

**Returns**

A pointer to the value at `--current`

This requires that `--current` is dereferenceable.

**operator[]()**

```
template<typename _Iterator>
reference std::reverse_iterator< _Iterator >::operator[] (
    difference_type __n) const [inline], [nodiscard], [constexpr]
```

**Returns**

The value at `current - __n - 1`

The underlying iterator must be a Random Access Iterator.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

**5.862 `__gnu_cxx::rope<_CharT, _Alloc>` Class Template Reference**

```
#include <rope>
```

**Public Types**

- typedef `_Rope_RopeConcatenation<_CharT, _Alloc>` `__C`
- typedef `_Rope_RopeFunction<_CharT, _Alloc>` `__F`
- typedef `_Rope_RopeLeaf<_CharT, _Alloc>` `__L`
- typedef `_Rope_RopeSubstring<_CharT, _Alloc>` `__S`
- typedef `__alloc_traits<_Alloc>::template rebind<__C>::other` `_CAlloc`
- typedef `__alloc_traits<_Alloc>::template rebind<_CharT>::other` `_DataAlloc`
- typedef `__alloc_traits<_Alloc>::template rebind<__F>::other` `_FAlloc`
- typedef `__alloc_traits<_Alloc>::template rebind<__L>::other` `_LAlloc`
- typedef `__alloc_traits<_Alloc>::template rebind<__S>::other` `_SAlloc`
- typedef `_Rope_const_iterator<_CharT, _Alloc>` `const_iterator`
- typedef `const _CharT *` `const_pointer`
- typedef `_CharT` `const_reference`
- typedef `std::reverse_iterator<const_iterator>` `const_reverse_iterator`
- typedef `std::ptrdiff_t` `difference_type`



- typedef `_Rope_iterator` < `_CharT`, `_Alloc` > **iterator**
- typedef `_Rope_char_ptr_proxy` < `_CharT`, `_Alloc` > **pointer**
- typedef `_Rope_char_ref_proxy` < `_CharT`, `_Alloc` > **reference**
- typedef `std::reverse_iterator` < `iterator` > **reverse\_iterator**
- typedef `std::size_t` **size\_type**
- typedef `_CharT` **value\_type**

## Public Member Functions

- **rope** (`_CharT` `__c`, `const allocator_type & __a=allocator_type()`)
- **rope** (`char_producer` < `_CharT` > `* __fn`, `size_type __len`, `bool __delete_fn`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const _CharT * __s`, `const _CharT * __e`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const _CharT * __s`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const _CharT * __s`, `size_type __len`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const allocator_type & __a=allocator_type()`)
- **rope** (`const const_iterator & __s`, `const const_iterator & __e`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const iterator & __s`, `const iterator & __e`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const rope & __x`, `const allocator_type & __a=allocator_type()`)
- **rope** (`size_type __n`, `_CharT __c`, `const allocator_type & __a=allocator_type()`)
- `allocator_type & _M_get_allocator ()`
- `const allocator_type & _M_get_allocator () const`
- **rope** & **append** ()
- **rope** & **append** (`_CharT __c`)
- **rope** & **append** (`const _CharT * __c_string`)
- **rope** & **append** (`const _CharT * __iter`, `size_type __n`)
- **rope** & **append** (`const _CharT * __s`, `const _CharT * __e`)
- **rope** & **append** (`const rope & __y`)
- **rope** & **append** (`const_iterator __s`, `const_iterator __e`)
- **rope** & **append** (`size_type __n`, `_CharT __c`)
- `void apply_to_pieces` (`size_type __begin`, `size_type __end`, `_Rope_char_consumer` < `_CharT` > & `__c`) `const`
- `_CharT at` (`size_type __pos`) `const`
- `_CharT back` () `const`
- `void balance` ()
- `const_iterator begin` ()
- `const_iterator begin` () `const`
- `const _CharT * c_str` () `const`
- `void clear` ()
- `int compare` (`const rope & __y`) `const`
- `const_iterator const_begin` () `const`
- `const_iterator const_end` () `const`
- `const_reverse_iterator const_rbegin` () `const`
- `const_reverse_iterator const_rend` () `const`
- `void copy` (`_CharT * __buffer`) `const`
- `size_type copy` (`size_type __pos`, `size_type __n`, `_CharT * __buffer`) `const`
- `void delete_c_str` ()
- `void dump` ()
- `bool empty` () `const`
- `const_iterator end` ()
- `const_iterator end` () `const`
- `iterator erase` (`const iterator & __p`)

- iterator **erase** (const iterator &\_\_p, const iterator &\_\_q)
- void **erase** (size\_type \_\_p, size\_type \_\_n)
- size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const
- size\_type **find** (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- \_CharT **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (const iterator &\_\_p)
- iterator **insert** (const iterator &\_\_p, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_type \_\_n)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*c\_string)
- iterator **insert** (const iterator &\_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const iterator &\_\_i, const iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const [rope](#) &\_\_r)
- iterator **insert** (const iterator &\_\_p, size\_type \_\_n, \_CharT \_\_c)
- void **insert** (size\_type \_\_p)
- void **insert** (size\_type \_\_p, \_CharT \_\_c)
- void **insert** (size\_type \_\_p, const \_CharT \*c\_string)
- void **insert** (size\_type \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **insert** (size\_type \_\_p, const \_CharT \*\_\_i, size\_type \_\_n)
- void **insert** (size\_type \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **insert** (size\_type \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- void **insert** (size\_type \_\_p, const [rope](#) &\_\_r)
- void **insert** (size\_type \_\_p, size\_type \_\_n, \_CharT \_\_c)
- size\_type **length** () const
- size\_type **max\_size** () const
- iterator **mutable\_begin** ()
- iterator **mutable\_end** ()
- [reverse\\_iterator](#) **mutable\_rbegin** ()
- reference **mutable\_reference\_at** (size\_type \_\_pos)
- [reverse\\_iterator](#) **mutable\_rend** ()
- [rope](#) & **operator=** (const [rope](#) &\_\_x)
- \_CharT **operator[]** (size\_type \_\_pos) const
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (\_CharT \_\_x)
- void **push\_front** (\_CharT \_\_x)
- [const\\_reverse\\_iterator](#) **rbegin** ()
- [const\\_reverse\\_iterator](#) **rbegin** () const
- [const\\_reverse\\_iterator](#) **rend** ()
- [const\\_reverse\\_iterator](#) **rend** () const
- void **replace** (const iterator &\_\_p, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const \_CharT \*c\_string)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_type \_\_n)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*c\_string)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, size\_type \_\_n)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const iterator &\_\_i, const iterator &\_\_j)

- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const rope &\_\_r)
- void **replace** (const iterator &\_\_p, const rope &\_\_r)
- void **replace** (const iterator &\_\_p, const\_iterator \_\_i, const\_iterator \_\_j)
- void **replace** (const iterator &\_\_p, iterator \_\_i, iterator \_\_j)
- void **replace** (size\_type \_\_p, \_CharT \_\_c)
- void **replace** (size\_type \_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (size\_type \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_type \_\_p, const \_CharT \*\_\_i, size\_type \_\_i\_len)
- void **replace** (size\_type \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_type \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (size\_type \_\_p, const rope &\_\_r)
- void **replace** (size\_type \_\_p, size\_type \_\_n, \_CharT \_\_c)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const \_CharT \*\_\_c\_string)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const \_CharT \*\_\_i, size\_type \_\_i\_len)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const rope &\_\_r)
- const \_CharT \* **replace\_with\_c\_str** ()
- size\_type **size** () const
- rope< \_CharT, \_Alloc > **substr** (const\_iterator \_\_start)
- rope **substr** (const\_iterator \_\_start, const\_iterator \_\_end) const
- rope **substr** (iterator \_\_start) const
- rope **substr** (iterator \_\_start, iterator \_\_end) const
- rope **substr** (size\_type \_\_start, size\_type \_\_len=1) const
- void **swap** (rope &\_\_b)

### Static Public Member Functions

- static \_\_C \* **\_C\_allocate** (std::size\_t \_\_n)
- static void **\_C\_deallocate** (\_\_C \*\_\_p, std::size\_t \_\_n)
- static \_CharT \* **\_Data\_allocate** (std::size\_t \_\_n)
- static void **\_Data\_deallocate** (\_CharT \*\_\_p, std::size\_t \_\_n)
- static \_\_F \* **\_F\_allocate** (std::size\_t \_\_n)
- static void **\_F\_deallocate** (\_\_F \*\_\_p, std::size\_t \_\_n)
- static \_\_L \* **\_L\_allocate** (std::size\_t \_\_n)
- static void **\_L\_deallocate** (\_\_L \*\_\_p, std::size\_t \_\_n)
- static \_\_S \* **\_S\_allocate** (std::size\_t \_\_n)
- static void **\_S\_deallocate** (\_\_S \*\_\_p, std::size\_t \_\_n)

### Public Attributes

- \_RopeRep \* **\_M\_tree\_ptr**

### Static Public Attributes

- static const size\_type **npos**

**Protected Types**

- enum { **\_S\_copy\_max** }
- typedef `_Rope_base<_CharT, _Alloc>` **\_Base**
- typedef `_CharT*` **\_Cstrptr**
- typedef `_Rope_RopeConcatenation<_CharT, _Alloc>` **\_RopeConcatenation**
- typedef `_Rope_RopeFunction<_CharT, _Alloc>` **\_RopeFunction**
- typedef `_Rope_RopeLeaf<_CharT, _Alloc>` **\_RopeLeaf**
- typedef `_Rope_RopeRep<_CharT, _Alloc>` **\_RopeRep**
- typedef `_Rope_RopeSubstring<_CharT, _Alloc>` **\_RopeSubstring**
- typedef `_Rope_self_destruct_ptr<_CharT, _Alloc>` **\_Self\_destruct\_ptr**
- typedef `_Base::allocator_type` **allocator\_type**

**Static Protected Member Functions**

- static `size_type` **\_S\_allocated\_capacity** (`size_type` \_\_n)
- static `bool` **\_S\_apply\_to\_pieces** (`_Rope_char_consumer<_CharT>` &\_\_c, `const _RopeRep*` \_\_r, `size_type` \_\_begin, `size_type` \_\_end)
- static `_RopeRep*` **\_S\_concat** (`_RopeRep*` \_\_left, `_RopeRep*` \_\_right)
- static `_RopeRep*` **\_S\_concat\_char\_iter** (`_RopeRep*` \_\_r, `const _CharT*` \_\_iter, `size_type` \_\_slen, `allocator_type` &\_\_a)
- static `_RopeRep*` **\_S\_destr\_concat\_char\_iter** (`_RopeRep*` \_\_r, `const _CharT*` \_\_iter, `size_type` \_\_slen, `allocator_type` &\_\_a)
- static `_RopeLeaf*` **\_S\_destr\_leaf\_concat\_char\_iter** (`_RopeLeaf*` \_\_r, `const _CharT*` \_\_iter, `size_type` \_\_slen)
- static `_CharT` **\_S\_fetch** (`_RopeRep*` \_\_r, `size_type` \_\_pos)
- static `_CharT*` **\_S\_fetch\_ptr** (`_RopeRep*` \_\_r, `size_type` \_\_pos)
- static `bool` **\_S\_is0** (`_CharT` \_\_c)
- static `_RopeLeaf*` **\_S\_leaf\_concat\_char\_iter** (`_RopeLeaf*` \_\_r, `const _CharT*` \_\_iter, `size_type` \_\_slen)
- static `_RopeConcatenation*` **\_S\_new\_RopeConcatenation** (`_RopeRep*` \_\_left, `_RopeRep*` \_\_right, `allocator_type` &\_\_a)
- static `_RopeFunction*` **\_S\_new\_RopeFunction** (`char_producer<_CharT>` \*\_\_f, `size_type` \_\_size, `bool` \_\_d, `allocator_type` &\_\_a)
- static `_RopeLeaf*` **\_S\_new\_RopeLeaf** (`_CharT*` \_\_s, `size_type` \_\_size, `allocator_type` &\_\_a)
- static `_RopeSubstring*` **\_S\_new\_RopeSubstring** (`_Rope_RopeRep<_CharT, _Alloc>` \*\_\_b, `size_type` \_\_s, `size_type` \_\_l, `allocator_type` &\_\_a)
- static `void` **\_S\_ref** (`_RopeRep*` \_\_t)
- static `_RopeLeaf*` **\_S\_RopeLeaf\_from\_unowned\_char\_ptr** (`const _CharT*` \_\_s, `size_type` \_\_size, `allocator_type` &\_\_a)
- static `size_type` **\_S\_rounded\_up\_size** (`size_type` \_\_n)
- static `_RopeRep*` **\_S\_substring** (`_RopeRep*` \_\_base, `size_type` \_\_start, `size_type` \_\_endp1)
- static `_RopeRep*` **\_S\_tree\_concat** (`_RopeRep*` \_\_left, `_RopeRep*` \_\_right)
- static `void` **\_S\_unref** (`_RopeRep*` \_\_t)
- static `_RopeRep*` **replace** (`_RopeRep*` \_\_old, `size_type` \_\_pos1, `size_type` \_\_pos2, `_RopeRep*` \_\_r)

**Static Protected Attributes**

- static `_CharT` **\_S\_empty\_c\_str** [1]

## Friends

- class `_Rope_char_ptr_proxy<_CharT, _Alloc>`
- class `_Rope_char_ref_proxy<_CharT, _Alloc>`
- class `_Rope_const_iterator<_CharT, _Alloc>`
- class `_Rope_iterator<_CharT, _Alloc>`
- class `_Rope_iterator_base<_CharT, _Alloc>`
- struct `_Rope_RopeRep<_CharT, _Alloc>`
- struct `_Rope_RopeSubstring<_CharT, _Alloc>`
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2> operator+ (const rope<_CharT2, _Alloc2> &__left, _CharT2 __right)`
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2> operator+ (const rope<_CharT2, _Alloc2> &__left, const _CharT2 *__right)`
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2> operator+ (const rope<_CharT2, _Alloc2> &__left, const rope<_CharT2, _Alloc2> &__right)`

### 5.862.1 Detailed Description

```
template<class _CharT, class _Alloc>
class __gnu_cxx::rope<_CharT, _Alloc>
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

The documentation for this class was generated from the following files:

- `rope`
- `ropeimpl.h`

## 5.863 std::runtime\_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for `std::runtime_error`:



## Public Member Functions

- `runtime_error (const char *)`
- `runtime_error (const runtime_error &) noexcept`

- `runtime_error` (const `string` &\_\_arg)
- `runtime_error` (`runtime_error` &&) noexcept
- `runtime_error` & `operator=` (const `runtime_error` &) noexcept
- `runtime_error` & `operator=` (`runtime_error` &&) noexcept
- virtual const char \* `what` () const noexcept

### 5.863.1 Detailed Description

One of two subclasses of exception.

Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

### 5.863.2 Constructor & Destructor Documentation

#### `runtime_error()`

```
std::runtime_error::runtime_error (
    const string & __arg) [explicit]
```

Takes a character string describing the error.

### 5.863.3 Member Function Documentation

#### `what()`

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from `std::exception`.

Reimplemented in `std::experimental::filesystem::v1::filesystem_error`, and `std::filesystem::filesystem_error`.

The documentation for this class was generated from the following file:

- `stdexcept`

## 5.864 `__gnu_pbds::sample_probe_fn` Class Reference

```
#include <sample_probe_fn.hpp>
```

### Public Types

- typedef std::size\_t `size_type`

### Public Member Functions

- `sample_probe_fn` ()
- `sample_probe_fn` (const `sample_probe_fn` &)
- void `swap` (`sample_probe_fn` &)

### Protected Member Functions

- `size_type` `operator()` (key\_const\_reference r\_key, `size_type` i) const

### 5.864.1 Detailed Description

A sample probe policy.

### 5.864.2 Constructor & Destructor Documentation

#### **sample\_probe\_fn()** [1/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn ()
```

Default constructor.

Referenced by [sample\\_probe\\_fn\(\)](#), and [swap\(\)](#).

#### **sample\_probe\_fn()** [2/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn (  
    const sample\_probe\_fn & )
```

Copy constructor.

References [sample\\_probe\\_fn\(\)](#).

### 5.864.3 Member Function Documentation

#### **operator>()()**

```
size_type __gnu_pbds::sample_probe_fn::operator() (  
    key_const_reference r_key,  
    size_type i) const [inline], [protected]
```

Returns the i-th offset from the hash value of some key r\_key.

#### **swap()**

```
void __gnu_pbds::sample_probe_fn::swap (  
    sample\_probe\_fn & ) [inline]
```

Swaps content.

References [sample\\_probe\\_fn\(\)](#).

The documentation for this class was generated from the following file:

- [sample\\_probe\\_fn.hpp](#)

## 5.865 \_\_gnu\_pbds::sample\_range\_hashing Class Reference

```
#include <sample_range_hashing.hpp>
```

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_range\\_hashing](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_range\\_hashing](#) &other)
- void [swap](#) ([sample\\_range\\_hashing](#) &other)

### Protected Member Functions

- void [notify\\_resized](#) ([size\\_type](#))
- [size\\_type](#) [operator\(\)](#) ([size\\_type](#)) const

### 5.865.1 Detailed Description

A sample range-hashing functor.

### 5.865.2 Member Typedef Documentation

#### `size_type`

`typedef std::size_t __gnu_pbds::sample_range_hashing::size_type`

Size type.

### 5.865.3 Constructor & Destructor Documentation

#### `sample_range_hashing()` [1/2]

`__gnu_pbds::sample_range_hashing::sample_range_hashing ()`

Default constructor.

Referenced by [sample\\_range\\_hashing\(\)](#), and [swap\(\)](#).

#### `sample_range_hashing()` [2/2]

`__gnu_pbds::sample_range_hashing::sample_range_hashing (  
const sample\_range\_hashing & other)`

Copy constructor.

References [sample\\_range\\_hashing\(\)](#).

### 5.865.4 Member Function Documentation

#### `notify_resized()`

`void __gnu_pbds::sample_range_hashing::notify_resized (  
size\_type ) [protected]`

Notifies the policy object that the container's size has changed to argument's size.

#### `operator>()`

`size\_type __gnu_pbds::sample_range_hashing::operator() (  
size\_type ) const [inline], [protected]`

Transforms the `__hash` value hash into a ranged-hash value.

#### `swap()`

`void __gnu_pbds::sample_range_hashing::swap (  
sample\_range\_hashing & other) [inline]`

Swaps content.

References [sample\\_range\\_hashing\(\)](#).

The documentation for this class was generated from the following file:

- [sample\\_range\\_hashing.hpp](#)

## 5.866 `__gnu_pbds::sample_ranged_hash_fn` Class Reference

`#include <sample_ranged_hash_fn.hpp>`

### Public Types

- `typedef std::size_t size_type`



## Public Member Functions

- [sample\\_ranged\\_hash\\_fn](#) ()
- [sample\\_ranged\\_hash\\_fn](#) (const [sample\\_ranged\\_hash\\_fn](#) &)
- void [swap](#) ([sample\\_ranged\\_hash\\_fn](#) &)

## Protected Member Functions

- void [notify\\_resized](#) (size\_type)
- size\_type [operator\(\)](#) (key\_const\_reference) const

### 5.866.1 Detailed Description

A sample ranged-hash functor.

### 5.866.2 Constructor & Destructor Documentation

#### [sample\\_ranged\\_hash\\_fn](#)() [1/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ()
```

Default constructor.

Referenced by [sample\\_ranged\\_hash\\_fn](#)(), and [swap](#)().

#### [sample\\_ranged\\_hash\\_fn](#)() [2/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn (  
    const sample\_ranged\_hash\_fn & )
```

Copy constructor.

References [sample\\_ranged\\_hash\\_fn](#)().

### 5.866.3 Member Function Documentation

#### [notify\\_resized](#)()

```
void __gnu_pbds::sample_ranged_hash_fn::notify_resized (  
    size_type ) [protected]
```

Notifies the policy object that the container's `__size` has changed to `size`.

#### [operator](#)()()

```
size_type __gnu_pbds::sample_ranged_hash_fn::operator() (  
    key_const_reference ) const [inline], [protected]
```

Transforms `key_const_reference` into a position within the table.

#### [swap](#)()

```
void __gnu_pbds::sample_ranged_hash_fn::swap (  
    sample\_ranged\_hash\_fn & ) [inline]
```

Swaps content.

References [sample\\_ranged\\_hash\\_fn](#)().

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_hash\\_fn.hpp](#)

## 5.867 \_\_gnu\_pbds::sample\_ranged\_probe\_fn Class Reference

```
#include <sample_ranged_probe_fn.hpp>
```

## Public Types

- typedef std::size\_t **size\_type**

## Public Member Functions

- **sample\_ranged\_probe\_fn** (const [sample\\_ranged\\_probe\\_fn](#) &)
- void **swap** ([sample\\_ranged\\_probe\\_fn](#) &)

## Protected Member Functions

- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference, std::size\_t, size\_type) const

### 5.867.1 Detailed Description

A sample ranged-probe functor.

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_probe\\_fn.hpp](#)

## 5.868 `__gnu_pbds::sample_resize_policy` Class Reference

```
#include <sample_resize_policy.hpp>
```

## Public Types

- typedef std::size\_t [size\\_type](#)

## Public Member Functions

- [sample\\_resize\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_policy](#) &other)
- void **swap** ([sample\\_resize\\_policy](#) &other)

## Protected Member Functions

- [size\\_type](#) **get\_new\_size** ([size\\_type](#) size, [size\\_type](#) num\_used\_e) const
- bool **is\_resize\_needed** () const
- void **notify\_cleared** ()
- void **notify\_erase\_search\_collision** ()
- void **notify\_erase\_search\_end** ()
- void **notify\_erase\_search\_start** ()
- void **notify\_erased** ([size\\_type](#) num\_e)
- void **notify\_find\_search\_collision** ()
- void **notify\_find\_search\_end** ()
- void **notify\_find\_search\_start** ()
- void **notify\_insert\_search\_collision** ()
- void **notify\_insert\_search\_end** ()
- void **notify\_insert\_search\_start** ()
- void **notify\_inserted** ([size\\_type](#) num\_e)
- void **notify\_resized** ([size\\_type](#) new\_size)

### 5.868.1 Detailed Description

A sample resize policy.

### 5.868.2 Member Typedef Documentation

#### **size\_type**

```
typedef std::size_t __gnu_pbds::sample_resize_policy::size_type
```

Size type.

### 5.868.3 Constructor & Destructor Documentation

#### **sample\_resize\_policy()**

```
__gnu_pbds::sample_resize_policy::sample_resize_policy ()
```

Default constructor.  
Referenced by [sample\\_range\\_hashing\(\)](#), and [swap\(\)](#).

### 5.868.4 Member Function Documentation

#### **get\_new\_size()**

```
size_type __gnu_pbds::sample_resize_policy::get_new_size (  
    size_type size,  
    size_type num_used_e) const [protected]
```

Queries what the new size should be.

#### **is\_resize\_needed()**

```
bool __gnu_pbds::sample_resize_policy::is_resize_needed () const [inline], [protected]
```

Queries whether a resize is needed.

#### **notify\_cleared()**

```
void __gnu_pbds::sample_resize_policy::notify_cleared () [protected]
```

Notifies the table was cleared.

#### **notify\_erase\_search\_collision()**

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

#### **notify\_erase\_search\_end()**

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_end () [inline], [protected]
```

Notifies a search ended.

#### **notify\_erase\_search\_start()**

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_start () [inline], [protected]
```

Notifies a search started.

#### **notify\_erased()**

```
void __gnu_pbds::sample_resize_policy::notify_erased (  
    size_type num_e) [inline], [protected]
```

Notifies an element was erased.

**notify\_find\_search\_collision()**

```
void __gnu_pbds::sample_resize_policy::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_find\_search\_end()**

```
void __gnu_pbds::sample_resize_policy::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_find\_search\_start()**

```
void __gnu_pbds::sample_resize_policy::notify_find_search_start () [inline], [protected]
```

Notifies a search started.

**notify\_insert\_search\_collision()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_insert\_search\_end()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_insert\_search\_start()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_start () [inline], [protected]
```

Notifies a search started.

**notify\_inserted()**

```
void __gnu_pbds::sample_resize_policy::notify_inserted (  
    size\_type num_e) [inline], [protected]
```

Notifies an element was inserted.

**notify\_resized()**

```
void __gnu_pbds::sample_resize_policy::notify_resized (  
    size\_type new_size) [protected]
```

Notifies the table was resized to new\_size.

**sample\_range\_hashing()**

```
__gnu_pbds::sample_resize_policy::sample_range_hashing (  
    const sample\_resize\_policy & other)
```

Copy constructor.

References [sample\\_resize\\_policy\(\)](#).

**swap()**

```
void __gnu_pbds::sample_resize_policy::swap (  
    sample\_resize\_policy & other) [inline]
```

Swaps content.

References [sample\\_resize\\_policy\(\)](#).

The documentation for this class was generated from the following file:

- [sample\\_resize\\_policy.hpp](#)

## 5.869 \_\_gnu\_pbds::sample\_resize\_trigger Class Reference

```
#include <sample_resize_trigger.hpp>
```

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_resize\\_trigger](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_trigger](#) &)
- void [swap](#) ([sample\\_resize\\_trigger](#) &)

### Protected Member Functions

- bool [is\\_grow\\_needed](#) ([size\\_type](#) size, [size\\_type](#) num\_entries) const
- bool [is\\_resize\\_needed](#) () const
- void [notify\\_cleared](#) ()
- void [notify\\_erase\\_search\\_collision](#) ()
- void [notify\\_erase\\_search\\_end](#) ()
- void [notify\\_erase\\_search\\_start](#) ()
- void [notify\\_erased](#) ([size\\_type](#) num\_entries)
- void [notify\\_externally\\_resized](#) ([size\\_type](#) new\_size)
- void [notify\\_find\\_search\\_collision](#) ()
- void [notify\\_find\\_search\\_end](#) ()
- void [notify\\_find\\_search\\_start](#) ()
- void [notify\\_insert\\_search\\_collision](#) ()
- void [notify\\_insert\\_search\\_end](#) ()
- void [notify\\_insert\\_search\\_start](#) ()
- void [notify\\_inserted](#) ([size\\_type](#) num\_entries)
- void [notify\\_resized](#) ([size\\_type](#) new\_size)

#### 5.869.1 Detailed Description

A sample resize trigger policy.

#### 5.869.2 Member Typedef Documentation

##### **size\_type**

```
typedef std::size_t __gnu_pbds::sample_resize_trigger::size_type
```

Size type.

#### 5.869.3 Constructor & Destructor Documentation

##### **sample\_resize\_trigger()**

```
__gnu_pbds::sample_resize_trigger::sample_resize_trigger ()
```

Default constructor.

Referenced by [sample\\_range\\_hashing\(\)](#), and [swap\(\)](#).

#### 5.869.4 Member Function Documentation

##### is\_grow\_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_grow_needed (
    size_type size,
    size_type num_entries) const [inline], [protected]
```

Queries whether a grow is needed.

##### is\_resize\_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_resize_needed () const [inline], [protected]
```

Queries whether a resize is needed.

##### notify\_cleared()

```
void __gnu_pbds::sample_resize_trigger::notify_cleared () [protected]
```

Notifies the table was cleared.

##### notify\_erase\_search\_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

##### notify\_erase\_search\_end()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_end () [inline], [protected]
```

Notifies a search ended.

##### notify\_erase\_search\_start()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_start () [inline], [protected]
```

Notifies a search started.

##### notify\_erased()

```
void __gnu_pbds::sample_resize_trigger::notify_erased (
    size_type num_entries) [inline], [protected]
```

Notifies an element was erased.

##### notify\_externally\_resized()

```
void __gnu_pbds::sample_resize_trigger::notify_externally_resized (
    size_type new_size) [protected]
```

Notifies the table was resized externally.

##### notify\_find\_search\_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

##### notify\_find\_search\_end()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_find\_search\_start()**

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_start () [inline], [protected]
```

Notifies a search started.

**notify\_insert\_search\_collision()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_insert\_search\_end()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_insert\_search\_start()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_start () [inline], [protected]
```

Notifies a search started.

**notify\_inserted()**

```
void __gnu_pbds::sample_resize_trigger::notify_inserted (
    size_type num_entries) [inline], [protected]
```

Notifies an element was inserted. the total number of entries in the table is num\_entries.

**notify\_resized()**

```
void __gnu_pbds::sample_resize_trigger::notify_resized (
    size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

**sample\_range\_hashing()**

```
__gnu_pbds::sample_resize_trigger::sample_range_hashing (
    const sample_resize_trigger & )
```

Copy constructor.  
References [sample\\_resize\\_trigger\(\)](#).

**swap()**

```
void __gnu_pbds::sample_resize_trigger::swap (
    sample_resize_trigger & ) [inline]
```

Swaps content.  
References [sample\\_resize\\_trigger\(\)](#).  
The documentation for this class was generated from the following file:

- [sample\\_resize\\_trigger.hpp](#)

**5.870 \_\_gnu\_pbds::sample\_size\_policy Class Reference**

```
#include <sample_size_policy.hpp>
```

**Public Types**

- typedef std::size\_t [size\\_type](#)

## Public Member Functions

- [sample\\_size\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_size\\_policy](#) &)
- void [swap](#) ([sample\\_size\\_policy](#) &other)

## Protected Member Functions

- [size\\_type](#) [get\\_nearest\\_larger\\_size](#) ([size\\_type](#) size) const
- [size\\_type](#) [get\\_nearest\\_smaller\\_size](#) ([size\\_type](#) size) const

### 5.870.1 Detailed Description

A sample size policy.

### 5.870.2 Member Typedef Documentation

#### `size_type`

typedef std::size\_t `__gnu_pbds::sample_size_policy::size_type`  
Size type.

### 5.870.3 Constructor & Destructor Documentation

#### `sample_size_policy()`

`__gnu_pbds::sample_size_policy::sample_size_policy` ()

Default constructor.

Referenced by [sample\\_range\\_hashing\(\)](#), and [swap\(\)](#).

### 5.870.4 Member Function Documentation

#### `get_nearest_larger_size()`

[size\\_type](#) `__gnu_pbds::sample_size_policy::get_nearest_larger_size` (  
    [size\\_type](#) size) const [inline], [protected]

Given a `__size` size, returns a `__size` that is larger.

#### `get_nearest_smaller_size()`

[size\\_type](#) `__gnu_pbds::sample_size_policy::get_nearest_smaller_size` (  
    [size\\_type](#) size) const [inline], [protected]

Given a `__size` size, returns a `__size` that is smaller.

#### `sample_range_hashing()`

`__gnu_pbds::sample_size_policy::sample_range_hashing` (  
    const [sample\\_size\\_policy](#) & )

Copy constructor.

References [sample\\_size\\_policy\(\)](#).

#### `swap()`

void `__gnu_pbds::sample_size_policy::swap` (  
    [sample\\_size\\_policy](#) & other) [inline]

Swaps content.

References [sample\\_size\\_policy\(\)](#).



The documentation for this class was generated from the following file:

- [sample\\_size\\_policy.hpp](#)

## 5.871 `__gnu_pbds::sample_tree_node_update`< `Const_Node_Iter`, `Node_Iter`, `Cmp_Fn`, `_Alloc` > Class Template Reference

```
#include <sample_tree_node_update.hpp>
```

### 5.871.1 Detailed Description

```
template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >
```

A sample node updatator.

The documentation for this class was generated from the following file:

- [sample\\_tree\\_node\\_update.hpp](#)

## 5.872 `__gnu_pbds::sample_trie_access_traits` Struct Reference

```
#include <sample_trie_access_traits.hpp>
```

### Public Types

- enum { `max_size` }
- typedef `std::string::const_iterator` `const_iterator`
- typedef char `e_type`
- typedef `rebind_traits< _Alloc, key\_type >::const_reference` `key_const_reference`
- typedef `std::string` `key_type`
- typedef `std::size_t` `size_type`

### Static Public Member Functions

- static `const_iterator` `begin` (`key_const_reference`)
- static `size_type` `e_pos` (`e_type`)
- static `const_iterator` `end` (`key_const_reference`)

### 5.872.1 Detailed Description

A sample trie element access traits.

### 5.872.2 Member Typedef Documentation

#### `e_type`

```
typedef char __gnu_pbds::sample_trie_access_traits::e_type
Element type.
```

### 5.872.3 Member Function Documentation

#### `begin()`

```
static const_iterator __gnu_pbds::sample_trie_access_traits::begin (
    key_const_reference ) [inline], [static]
```

Returns a `const_iterator` to the first element of `r_key`.

**e\_pos()**

```
static size_type __gnu_pbds::sample_trie_access_traits::e_pos (
    e_type ) [inline], [static]
```

Maps an element to a position.

**end()**

```
static const_iterator __gnu_pbds::sample_trie_access_traits::end (
    key_const_reference ) [inline], [static]
```

Returns a const\_iterator to the after-last element of r\_key.

The documentation for this struct was generated from the following file:

- [sample\\_trie\\_access\\_traits.hpp](#)

## 5.873 `__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

```
#include <sample_trie_node_update.hpp>
```

**Public Types**

- typedef std::size\_t **metadata\_type**

**Protected Member Functions**

- [sample\\_trie\\_node\\_update](#) ()
- void [operator](#)() (node\_iterator, node\_const\_iterator) const

**5.873.1 Detailed Description**

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A sample node updator.

**5.873.2 Constructor & Destructor Documentation****sample\_trie\_node\_update()**

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_↵
update () [protected]
```

Default constructor.

**5.873.3 Member Function Documentation****operator>()()**

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (
    node_iterator ,
    node_const_iterator ) const [inline], [protected]
```

Updates the rank of a node through a node\_iterator node\_it; end\_nd\_it is the end node iterator.

The documentation for this class was generated from the following file:

- [sample\\_trie\\_node\\_update.hpp](#)

## 5.874 `__gnu_pbds::sample_update_policy` Struct Reference

```
#include <sample_update_policy.hpp>
```

### Public Member Functions

- [sample\\_update\\_policy](#) ()
- [sample\\_update\\_policy](#) (const [sample\\_update\\_policy](#) &)
- void [swap](#) ([sample\\_update\\_policy](#) &other)

### Protected Types

- typedef some\_metadata\_type [metadata\\_type](#)

### Protected Member Functions

- [metadata\\_type operator\(\)](#) () const
- bool [operator\(\)](#) (metadata\_reference) const

#### 5.874.1 Detailed Description

A sample list-update policy.

#### 5.874.2 Member Typedef Documentation

##### `metadata_type`

```
typedef some_metadata_type __gnu_pbds::sample_update_policy::metadata_type [protected]
```

Metadata on which this functor operates.

#### 5.874.3 Constructor & Destructor Documentation

##### `sample_update_policy()` [1/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy ()
```

Default constructor.

Referenced by [sample\\_update\\_policy\(\)](#), and [swap\(\)](#).

##### `sample_update_policy()` [2/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy (  
    const sample\_update\_policy & )
```

Copy constructor.

References [sample\\_update\\_policy\(\)](#).

#### 5.874.4 Member Function Documentation

##### `operator>()()` [1/2]

```
metadata\_type __gnu_pbds::sample_update_policy::operator() () const [protected]
```

Creates a metadata object.

**operator()()** [2/2]

```
bool __gnu_pbds::sample_update_policy::operator() (
    metadata_reference ) const [protected]
```

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method should return true if the node should be moved to the front of the list.

**swap()**

```
void __gnu_pbds::sample_update_policy::swap (
    sample_update_policy & other) [inline]
```

Swaps content.

References [sample\\_update\\_policy\(\)](#).

The documentation for this struct was generated from the following file:

- [sample\\_update\\_policy.hpp](#)

**5.875 \_\_gnu\_parallel::sampling\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for \_\_gnu\_parallel::sampling\_tag:

**Public Member Functions**

- [sampling\\_tag](#) ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

**5.875.1 Detailed Description**

Forces parallel merging with exact splitting, at compile time.

**5.875.2 Member Function Documentation****\_\_get\_num\_threads()**

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

**Returns**

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

**set\_num\_threads()**

```
void __gnu_parallel::parallel_tag::set_num_threads (
    __ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

**Parameters**

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.876 `std::scoped_allocator_adaptor<_OuterAlloc, _InnerAllocs >` Class Template Reference

```
#include <scoped_allocator>
```

**Public Types**

- typedef [\\_\\_traits::const\\_pointer](#) **const\_pointer**
- typedef [\\_\\_traits::const\\_void\\_pointer](#) **const\_void\_pointer**
- typedef [\\_\\_traits::difference\\_type](#) **difference\_type**
- typedef [\\_\\_inner\\_type::\\_\\_type](#) **inner\_allocator\_type**
- typedef [\\_\\_and< typename \\_\\_traits::is\\_always\\_equal, typename allocator\\_traits< \\_InnerAllocs >::is\\_always\\_equal... >::type](#) **is\_always\_equal**
- typedef `_OuterAlloc` **outer\_allocator\_type**
- typedef [\\_\\_traits::pointer](#) **pointer**
- typedef [\\_\\_or< typename \\_\\_traits::propagate\\_on\\_container\\_copy\\_assignment, typename allocator\\_traits< \\_InnerAllocs >::propagate\\_on\\_container\\_copy\\_assignment... >::type](#) **propagate\_on\_container\_copy\_assignment**
- typedef [\\_\\_or< typename \\_\\_traits::propagate\\_on\\_container\\_move\\_assignment, typename allocator\\_traits< \\_InnerAllocs >::propagate\\_on\\_container\\_move\\_assignment... >::type](#) **propagate\_on\_container\_move\_assignment**
- typedef [\\_\\_or< typename \\_\\_traits::propagate\\_on\\_container\\_swap, typename allocator\\_traits< \\_InnerAllocs >::propagate\\_on\\_container\\_swap... >::type](#) **propagate\_on\_container\_swap**
- typedef [\\_\\_traits::size\\_type](#) **size\_type**
- typedef [\\_\\_traits::value\\_type](#) **value\_type**
- typedef [\\_\\_traits::void\\_pointer](#) **void\_pointer**

**Public Member Functions**

- `template<typename _Outer2, typename = _Constructible<_Outer2>>>`  
**scoped\_allocator\_adaptor** (`_Outer2` &&\_\_outer, const `_InnerAllocs` &... \_\_inner) noexcept
- **scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#) &\_\_other) noexcept
- `template<typename _Outer2, typename = _Constructible<const _Outer2>>>`  
**scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#)< `_Outer2`, `_InnerAllocs`... > &\_\_other) noexcept

- **scoped\_allocator\_adaptor** ([scoped\\_allocator\\_adaptor](#) &&\_\_other) noexcept
- template<typename \_Outer2, typename = \_Constructible<\_Outer2>>  
**scoped\_allocator\_adaptor** ([scoped\\_allocator\\_adaptor](#)< \_Outer2, \_InnerAllocs... > &&\_\_other) noexcept
- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, const\_void\_pointer \_\_hint)
- template<typename \_Tp, typename... \_Args>  
void **construct** (\_Tp \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n) noexcept
- template<typename \_Tp>  
void **destroy** (\_Tp \*\_\_p)
- const inner\_allocator\_type & **inner\_allocator** () const noexcept
- inner\_allocator\_type & **inner\_allocator** () noexcept
- size\_type **max\_size** () const
- [scoped\\_allocator\\_adaptor](#) & **operator=** (const [scoped\\_allocator\\_adaptor](#) &)=default
- [scoped\\_allocator\\_adaptor](#) & **operator=** ([scoped\\_allocator\\_adaptor](#) &&)=default
- const outer\_allocator\_type & **outer\_allocator** () const noexcept
- outer\_allocator\_type & **outer\_allocator** () noexcept
- [scoped\\_allocator\\_adaptor](#) **select\_on\_container\_copy\_construction** () const

## Friends

- template<typename...>  
struct **\_\_inner\_type\_impl**
- template<typename \_OutA1, typename \_OutA2, typename... \_InA>  
bool **operator==** (const [scoped\\_allocator\\_adaptor](#)< \_OutA1, \_InA... > &\_\_a, const [scoped\\_allocator\\_adaptor](#)< \_OutA2, \_InA... > &\_\_b) noexcept

## Related Symbols

(Note that these are not member symbols.)

- template<typename \_OutA1, typename \_OutA2, typename... \_InA>  
bool **operator==** (const [scoped\\_allocator\\_adaptor](#)< \_OutA1, \_InA... > &\_\_a, const [scoped\\_allocator\\_adaptor](#)< \_OutA2, \_InA... > &\_\_b) noexcept

### 5.876.1 Detailed Description

```
template<typename _OuterAlloc, typename... _InnerAllocs>
class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >
```

An adaptor to recursively pass an allocator to the objects it constructs.  
The documentation for this class was generated from the following file:

- [scoped\\_allocator](#)

## 5.877 std::seed\_seq Class Reference

```
#include <random>
```

### Public Types

- typedef uint\_least32\_t [result\\_type](#)

## Public Member Functions

- [seed\\_seq](#) () noexcept
- `template<typename _InputIterator>`  
`seed_seq` (\_InputIterator \_\_begin, \_InputIterator \_\_end)
- `seed_seq` (const [seed\\_seq](#) &)=delete
- `template<typename _IntType, typename = _Require<is_integral<_IntType>>>`  
`seed_seq` (std::initializer\_list< \_IntType > \_\_il)
- `template<typename _RandomAccessIterator>`  
void **generate** (\_RandomAccessIterator \_\_begin, \_RandomAccessIterator \_\_end)
- `seed_seq` & **operator=** (const [seed\\_seq](#) &)=delete
- `template<typename _OutputIterator>`  
void **param** (\_OutputIterator \_\_dest) const
- `size_t` **size** () const noexcept

### 5.877.1 Detailed Description

The `seed_seq` class generates sequences of seeds for random number generators.

Since

C++11

### 5.877.2 Member Typedef Documentation

#### **result\_type**

```
typedef uint_least32_t std::seed\_seq::result\_type
```

The type of the seed vales.

### 5.877.3 Constructor & Destructor Documentation

#### **seed\_seq()**

```
std::seed_seq::seed_seq () [inline], [noexcept]
```

Default constructor.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.878 \_\_gnu\_cxx::select1st< \_Pair > Struct Template Reference

```
#include <functional>
```

## Public Types

- typedef \_Pair [argument\\_type](#)
- typedef \_Pair::first\_type [result\\_type](#)

## Public Member Functions

- \_Pair::first\_type & **operator()** (\_Pair &\_\_x) const
- `template<typename _Pair2>`  
\_Pair2::first\_type & **operator()** (\_Pair2 &\_\_x) const
- const \_Pair::first\_type & **operator()** (const \_Pair &\_\_x) const
- `template<typename _Pair2>`  
const \_Pair2::first\_type & **operator()** (const \_Pair2 &\_\_x) const

### 5.878.1 Detailed Description

```
template<class _Pair>
struct __gnu_cxx::select1st<_Pair>
```

An [SGI extension](#) .

### 5.878.2 Member Typedef Documentation

#### argument\_type

```
typedef _Pair std::unary_function<_Pair, _Pair::first_type>::argument_type [inherited]
argument_type is the type of the argument
```

#### result\_type

```
typedef _Pair::first_type std::unary_function<_Pair, _Pair::first_type>::result_type [inherited]
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.879 `__gnu_cxx::select2nd<_Pair>` Struct Template Reference

```
#include <functional>
```

### Public Types

- typedef \_Pair [argument\\_type](#)
- typedef \_Pair::second\_type [result\\_type](#)

### Public Member Functions

- \_Pair::second\_type & **operator()** (\_Pair &\_\_x) const
- const \_Pair::second\_type & **operator()** (const \_Pair &\_\_x) const

### 5.879.1 Detailed Description

```
template<class _Pair>
struct __gnu_cxx::select2nd<_Pair>
```

An [SGI extension](#) .

### 5.879.2 Member Typedef Documentation

#### argument\_type

```
typedef _Pair std::unary_function<_Pair, _Pair::second_type>::argument_type [inherited]
argument_type is the type of the argument
```

#### result\_type

```
typedef _Pair::second_type std::unary_function<_Pair, _Pair::second_type>::result_type [inherited]
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [ext/functional](#)



### 5.880 `__gnu_pbds::detail::select_value_type< Key, Mapped >` Struct Template Reference

```
#include <types_traits.hpp>
```

#### Public Types

- typedef `std::pair< const Key, Mapped >` **type**

#### 5.880.1 Detailed Description

```
template<typename Key, typename Mapped>
struct __gnu_pbds::detail::select_value_type< Key, Mapped >
```

Choose `value_type` to be a key/value pair or just a key.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

### 5.881 `__gnu_pbds::detail::select_value_type< Key, null_type >` Struct Template Reference

```
#include <types_traits.hpp>
```

#### Public Types

- typedef `std::pair< const Key, null_type >` **type**
- typedef `Key` **type**

#### 5.881.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::select_value_type< Key, null_type >
```

Specialization for sets where the key is the `value_type`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

### 5.882 `std::basic_istream< _CharT, _Traits >::sentry` Class Reference

```
#include <istream>
```

Inheritance diagram for std::basic\_istream< \_CharT, \_Traits >::sentry:



## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
- typedef [\\_\\_istream\\_type::\\_\\_ctype\\_type](#) **\_\_ctype\_type**
- typedef \_Traits::int\_type **\_\_int\_type**
- typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
- typedef [basic\\_istream](#)< \_CharT, \_Traits > **\_\_istream\_type**
- typedef [basic\\_istream](#)< \_CharT, \_Traits > **\_\_istream\_type**
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**
- typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
- typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum [event](#)
- typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_Traits::int\_type **int\_type**
- typedef [\\_ios\\_istate](#) **istate**
- typedef \_Traits::off\_type **off\_type**
- typedef [\\_ios\\_Openmode](#) **openmode**
- typedef \_Traits::pos\_type **pos\_type**
- typedef [\\_ios\\_Seekdir](#) **seekdir**
- typedef \_Traits **traits\_type**
- typedef [\\_Traits](#) **traits\_type**
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**

## Public Member Functions

- [sentry](#) ([basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > &\_\_is, bool \_\_noskipws=false)
- virtual [~basic\\_istream](#) ()
- [basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > & [M\\_extract](#) ([\\_ValueT](#) &\_\_v)
- [basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > & [M\\_extract](#) ([\\_ValueT](#) &\_\_v)
- const [locale](#) & [M\\_getloc](#) () const
- void [M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- [basic\\_istream](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize](#) [gcount](#) () const
- [basic\\_istream](#)< [char](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [basic\\_istream](#)< [char](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [basic\\_istream](#)< [wchar\\_t](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [basic\\_istream](#)< [wchar\\_t](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n)
- [basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)
- [char](#) [narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_default) const
- [operator bool](#) () const
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- [\\_\\_istream\\_type](#) & [operator>>](#) (void \*&\_\_p)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) () const
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- [iostate](#) [rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [fmtflags](#) [self](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [self](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [setstate](#) ([iostate](#) \_\_state)

- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \* \_\_tiestr)
- void [unsetf](#) (fmtflags \_\_mask)
- char\_type [widen](#) (char \_\_c) const
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_istream\\_type](#) &(\*\_\_pf)([\\_\\_istream\\_type](#) &))
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
- [\\_\\_istream\\_type](#) & [operator>>](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Extractors

All the *operator>>* functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (*noskipws*) set to false. This has several effects, concluding with the setting of a status flag; see the *sentry* documentation for more.

If the *sentry* status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- [\\_\\_istream\\_type](#) & [operator>>](#) (bool &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (short &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned short &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (int &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned int &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (float &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (double &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (*noskipws*) set to true. This has several effects, concluding with the setting of a status flag; see the *sentry* documentation for more.

If the *sentry* status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- int\_type [get](#) ()
- [\\_\\_istream\\_type](#) & [get](#) (char\_type &\_\_c)
- [\\_\\_istream\\_type](#) & [get](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) & \_\_sb, char\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) & \_\_sb)
- [\\_\\_istream\\_type](#) & [getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)

- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ()
  - [int\\_type](#) [peek](#) ()
  - [\\_\\_istream\\_type](#) & [read](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
  - [streamsize](#) [readsome](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [putback](#) ([char\\_type](#) \_\_c)
  - [\\_\\_istream\\_type](#) & [unget](#) ()
  - [int](#) [sync](#) ()
  - [pos\\_type](#) [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) ([pos\\_type](#))
  - [\\_\\_istream\\_type](#) & [seekg](#) ([off\\_type](#), [ios\\_base::seekdir](#))
- 
- [operator bool](#) () const
  - [bool](#) [operator!](#) () const

### Static Public Member Functions

- static [bool](#) [sync\\_with\\_stdio](#) ([bool](#) \_\_sync=true)
- static [int](#) [xalloc](#) () throw ()

### Public Attributes

- class [\\_\\_attribute\\_\\_\(\(\\_\\_abi\\_tag\\_\\_\("cxx11"\)\)\)](#) failure typedef [\\_ios\\_Fmtflags](#) [fmtflags](#)

### Static Public Attributes

- static const [openmode](#) [\\_\\_noreplace](#)
- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iosstate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iosstate](#) [eofbit](#)
- static const [iosstate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iosstate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)

- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

### Protected Types

- enum

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) & \_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_\\_istream\\_type](#) & [\\_M\\_extract](#) (\_ValueT & \_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) & \_\_rhs) noexcept
- [basic\\_istream](#) ()
- [basic\\_istream](#) ([basic\\_istream](#) && \_\_rhs)
- [basic\\_istream](#) (const [basic\\_istream](#) &)=delete
- void [init](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
- void [move](#) ([basic\\_ios](#) && \_\_rhs)
- void [move](#) ([basic\\_ios](#) & \_\_rhs)
- [basic\\_istream](#) & [operator=](#) ([basic\\_istream](#) && \_\_rhs)
- [basic\\_istream](#) & [operator=](#) (const [basic\\_istream](#) &)=delete
- void [set\\_rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
- void [swap](#) ([basic\\_ios](#) & \_\_rhs) noexcept
- void [swap](#) ([basic\\_istream](#) & \_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const [\\_\\_ctype\\_type](#) \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- [char\\_type](#) [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [streamsize](#) [\\_M\\_gcount](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* [\\_M\\_num\\_get](#)
- const [\\_\\_num\\_put\\_type](#) \* [\\_M\\_num\\_put](#)
- [streamsize](#) [\\_M\\_precision](#)
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [\\_M\\_streambuf](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

### 5.882.1 Detailed Description

**template<typename \_CharT, typename \_Traits>**  
**class std::basic\_istream< \_CharT, \_Traits >::sentry**

Performs setup work for input streams.

Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations*, although only prefix actions are currently required by the standard.

### 5.882.2 Member Typedef Documentation

#### **\_\_num\_put\_type**

```
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits  
>::__num_put_type
```

These are non-standard types.

#### **event\_callback**

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)
```

The type of an event callback function.

#### Parameters

|                                                                                                |                                                        |
|------------------------------------------------------------------------------------------------|--------------------------------------------------------|
|  <b>__e</b>   | One of the members of the event enum.                  |
|  <b>__b</b>   | Reference to the ios_base object.                      |
|  <b>__i</b> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

#### **iostate**

```
typedef __Ios_Iostate std::ios_base::iostate
```

This is a bitmask type.

`__Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

#### **openmode**

```
typedef __Ios_Openmode std::ios_base::openmode
```

This is a bitmask type.

`__Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app

- ate
- binary
- in
- out
- trunc

### seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

### traits\_type

```
template<typename _CharT, typename _Traits>
```

```
typedef _Traits std::basic_istream<_CharT, _Traits>::sentry::traits_type
```

Easy access to dependent types.

## 5.882.3 Member Enumeration Documentation

### event

```
enum std::ios_base::event
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

## 5.882.4 Constructor & Destructor Documentation

### sentry()

```
template<typename _CharT, typename _Traits>
```

```
std::basic_istream<_CharT, _Traits>::sentry::sentry (
    basic_istream<_CharT, _Traits> & __is,
    bool __noskipws = false) [explicit]
```

The constructor performs all the work.

#### Parameters

|                         |                                       |
|-------------------------|---------------------------------------|
| <code>__is</code>       | The input stream to guard.            |
| <code>__noskipws</code> | Whether to consume whitespace or not. |

If the stream state is good (`__is.good()` is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.

The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence



2. if `__noskipws` is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

If the stream state is still good, then the sentry state becomes true (*okay*).

References [std::ios\\_base::badbit](#), [basic\\_istream\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::good\(\)](#), and [std::ios\\_base::goodbit](#).

### **~basic\_istream()**

```
virtual std::basic_istream<_CharT, _Traits>::~~basic_istream () [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

## **5.882.5 Member Function Documentation**

### **\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline]
```

Locale access.

#### **Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

### **bad()**

```
bool std::basic_ios<_CharT, _Traits>::bad () const [inline], [nodiscard]
```

Fast error checking.

#### **Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

### **basic\_istream()**

```
std::basic_istream<_CharT, _Traits>::basic_istream (  
    __streambuf_type * __sb) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Referenced by [sentry\(\)](#).

### **clear()**

```
void std::basic_ios<_CharT, _Traits>::clear (  
    iostate __state = goodbit)
```

[Re]sets the error state.

#### **Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

**copyfmt()**

```
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
    const basic_ios & __rhs)
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

**eof()**

```
bool std::basic_ios<_CharT, _Traits>::eof () const [inline], [nodiscard]
Fast error checking.
```

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

**exceptions() [1/2]**

```
iostate std::basic_ios<_CharT, _Traits>::exceptions () const [inline], [nodiscard]
Throwing exceptions on errors.
```

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

**exceptions() [2/2]**

```
void std::basic_ios<_CharT, _Traits>::exceptions (
    iostate __except) [inline]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
```

```
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

### fail()

bool `std::basic_ios< _CharT, _Traits >::fail ()` const [inline], [nodiscard]  
Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

### fill() [1/2]

char\_type `std::basic_ios< _CharT, _Traits >::fill ()` const [inline], [nodiscard]  
Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

### fill() [2/2]

char\_type `std::basic_ios< _CharT, _Traits >::fill (`  
    char\_type \_\_ch) [inline]

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

### flags() [1/2]

fmtflags `std::ios_base::flags ()` const [inline], [nodiscard]  
Access to format flags.

#### Returns

The format control flags for both input and output.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**gcount()**

```
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
    void )
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

**get()** [2/6]

```
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb) [inline]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

**get()** [3/6]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim)
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

**get()** [4/6]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c)
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

**get()** [5/6]

```
__istream_type & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n) [inline]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                |
| <code>__n</code> | Maximum number of characters to store in <i>s</i> . |

**Returns**

\*this

Returns `get(__s,__n,widen("\n"))`.

**get()** [6/6]

```
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
    char_type * __s,
    streamsize __n,
    char_type __delim)
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

**getline()** [1/4]

```
__istream_type & std::basic_istream<_CharT, _Traits>::getline (
    char_type * __s,
    streamsize __n) [inline]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s, __n, widen("\n"))`.

**getline()** [2/4]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (  
    char_type * __s,  
    streamsize __n,  
    char_type __delim)
```

String extraction.

## Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

## Returns

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.) In any case, a null character is stored in the next location in the array.

**getline()** [3/4]

```
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim)
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getline()** [4/4]

```
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim)
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard]
```

Locale access.

## Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

**good()**

```
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard]
```

Fast error checking.

## Returns

True if no error flags are set.

A wrapper around `rdstate`.



**ignore()** [1/3]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    void )
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

**ignore()** [2/3]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n)
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

**ignore()** [3/3]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim)
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

**imbue()**

```
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc)
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↵html>

**init()**

```
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb) [protected]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline]
```

Access to integer array.

**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>↵<br/>__ix</code> | Index into the array. |
|-------------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**narrow()**

```
char std::basic_ios< _CharT, _Traits >::narrow (
    char_type __c,
    char __default) const [inline]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool() [1/2]**

```
std::basic_ios<_CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator bool() [2/2]**

```
template<typename _CharT, typename _Traits>
```

```
std::basic_istream<_CharT, _Traits >::sentry::operator bool () const [inline], [explicit]
```

Quick status checking.

**Returns**

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

**operator!()**

```
bool std::basic_ios<_CharT, _Traits >::operator! () const [inline], [nodiscard]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator>>() [1/17]**

```
__istream_type & std::basic_istream<_CharT, _Traits >::operator>> (  
    __ios_type &(* __pf) (__ios_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.

For more information, see the `io manip` header.

**operator>>() [2/17]**

```
__istream_type & std::basic_istream<_CharT, _Traits >::operator>> (  
    __istream_type &(* __pf) (__istream_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.

For more information, see the `io manip` header.

**operator>>() [3/17]**

```
basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> (  
    __streambuf_type * __sb)
```

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

**operator>>()** [4/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n) [inline]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [5/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    double & __f) [inline]
```

Floating point arithmetic extractors.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [6/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    float & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| <i>f</i>        |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [7/17]**

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n)
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| <i>n</i>        |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [8/17]**

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    ios_base &(* __pf ) (ios_base &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>() [9/17]**

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| <i>n</i>        |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [10/17]

```
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long double & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [11/17]

```
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    long long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↩        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [12/17]

```
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
    short & __n)
```

Integer arithmetic extractors.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↩        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [13/17]

```
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned int & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [14/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (  
    unsigned long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [15/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (  
    unsigned long long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [16/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (  
    unsigned short & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [17/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p) [inline]
```

Basic arithmetic extractors.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void )
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec) [inline]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.



**putback()**

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c)
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf() [1/2]**

```
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

**rdbuf() [2/2]**

```
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb)
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

**rdstate()**

```
iosstate std::basic_ios<_CharT, _Traits>::rdstate() const [inline], [nodiscard]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

**read()**

```
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read (
    char_type * __s,
    streamsize __n)
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

`*this`

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

**readsome()**

```
streamsize std::basic_istream< _CharT, _Traits >::readsome (
    char_type * __s,
    streamsize __n)
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rddbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index)
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**seekg() [1/2]**

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    off_type __off,
    ios_base::seekdir __dir)
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

**seekg()** [2/2]

```
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
    pos_type __pos)
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

**setstate()**

```
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state) [inline]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

**sync()**

```
int std::basic_istream< _CharT, _Traits >::sync (
    void )
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void )
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

**tie()** [1/2]

```
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

**tie()** [2/2]

```
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr) [inline]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
    void )
```

Unextracting the previous character.

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c) const [inline]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**width() [1/2]**

```
streamsize std::ios_base::width () const [inline], [nodiscard]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

**width() [2/2]**

```
streamsize std::ios_base::width (
    streamsize __wide) [inline]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.882.6 Member Data Documentation****M\_gcount**

```
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

**adjustfield**

```
const fmtflags std::ios_base::adjustfield [static]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

**app**

```
const openmode std::ios_base::app [static]
```

Seek to end before each write.

**ate**

```
const openmode std::ios_base::ate [static]
```

Open and seek to end immediately after opening.

**badbit**

```
const iostate std::ios_base::badbit [static]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

**basefield**

```
const fmtflags std::ios_base::basefield [static]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

**beg**

```
const seekdir std::ios_base::beg [static]
```

Request a seek relative to the beginning of the stream.

**binary**

```
const openmode std::ios_base::binary [static]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary>.



**boolalpha**

const `fmtflags` `std::ios_base::boolalpha` [static]  
Insert/extract `bool` in alphabetic rather than numeric format.

**cur**

const `seekdir` `std::ios_base::cur` [static]  
Request a seek relative to the current position within the sequence.

**dec**

const `fmtflags` `std::ios_base::dec` [static]  
Converts integer input or generates integer output in decimal base.

**end**

const `seekdir` `std::ios_base::end` [static]  
Request a seek relative to the current end of the sequence.

**eofbit**

const `istate` `std::ios_base::eofbit` [static]  
Indicates that an input operation reached the end of an input sequence.

**failbit**

const `istate` `std::ios_base::failbit` [static]  
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

**fixed**

const `fmtflags` `std::ios_base::fixed` [static]  
Generate floating-point output in fixed-point notation.

**floatfield**

const `fmtflags` `std::ios_base::floatfield` [static]  
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

**fmtflags**

class \_\_attribute\_\_((\_\_abi\_tag\_\_ ("cxx11"))) failure typedef `_Ios_Fmtflags` `std::ios_base::fmtflags`  
These are thrown to indicate problems with io.

**27.4.2.1.1 Class `ios_base::failure`**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`

- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

### goodbit

const `ios_base::goodbit` [static]

Indicates all is well.

### hex

const `ios_base::hex` [static]

Converts integer input or generates integer output in hexadecimal base.

### in

const `ios_base::in` [static]

Open for input. Default for `ifstream` and `fstream`.

### internal

const `ios_base::internal` [static]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

### left

const `ios_base::left` [static]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

### oct

const `ios_base::oct` [static]

Converts integer input or generates integer output in octal base.

**out**

`const openmode std::ios_base::out [static]`  
Open for output. Default for `ofstream` and `fstream`.

**right**

`const fmtflags std::ios_base::right [static]`  
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

**scientific**

`const fmtflags std::ios_base::scientific [static]`  
Generates floating-point output in scientific notation.

**showbase**

`const fmtflags std::ios_base::showbase [static]`  
Generates a prefix indicating the numeric base of generated integer output.

**showpoint**

`const fmtflags std::ios_base::showpoint [static]`  
Generates a decimal-point character unconditionally in generated floating-point output.

**showpos**

`const fmtflags std::ios_base::showpos [static]`  
Generates a + sign in non-negative generated numeric output.

**skipws**

`const fmtflags std::ios_base::skipws [static]`  
Skips leading white space before certain input operations.

**trunc**

`const openmode std::ios_base::trunc [static]`  
Truncate an existing stream when opening. Default for `ofstream`.

**unitbuf**

`const fmtflags std::ios_base::unitbuf [static]`  
Flushes output after each output operation.

**uppercase**

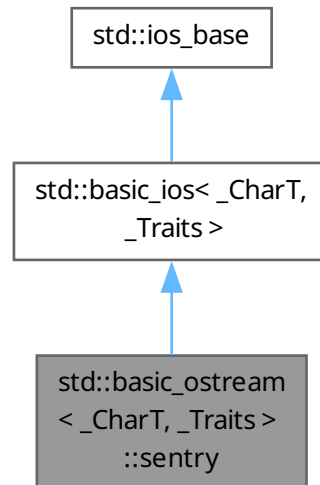
`const fmtflags std::ios_base::uppercase [static]`  
Replaces certain lowercase letters with their uppercase equivalents in generated output.  
The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

## 5.883 std::basic\_ostream< \_CharT, \_Traits >::sentry Class Reference

```
#include <ostream.h>
```

Inheritance diagram for std::basic\_ostream< \_CharT, \_Traits >::sentry:



### Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
- typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**
- typedef [basic\\_ostream](#)< \_CharT, \_Traits > **\_\_ostream\_type**
- typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum [event](#)
- typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_Traits::int\_type **int\_type**
- typedef \_ios\_istate [iostate](#)
- typedef \_Traits::off\_type **off\_type**
- typedef \_ios\_Openmode [openmode](#)
- typedef \_Traits::pos\_type **pos\_type**
- typedef \_ios\_Seekdir [seekdir](#)
- typedef \_Traits **traits\_type**
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**

### Public Member Functions

- [sentry](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os)
- virtual [~basic\\_ostream](#) ()

- `~sentry ()`
  - `const locale & _M_getloc () const`
  - `basic_ostream< _CharT, _Traits > & _M_insert (_ValueT __v)`
  - `basic_ostream< _CharT, _Traits > & _M_insert (_ValueT __v)`
  - `void _M_setstate (iostate __state)`
  - `bool bad () const`
  - `basic_ostream (__streambuf_type *__sb)`
  - `void clear (iostate __state=goodbit)`
  - `basic_ios & copyfmt (const basic_ios & __rhs)`
  - `bool eof () const`
  - `iostate exceptions () const`
  - `void exceptions (iostate __except)`
  - `bool fail () const`
  - `char_type fill () const`
  - `char_type fill (char_type __ch)`
  - `fmtflags flags () const`
  - `fmtflags flags (fmtflags __fmtfl)`
  - `__ostream_type & flush ()`
  - `locale getloc () const`
  - `bool good () const`
  - `locale imbue (const locale & __loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `operator bool () const`
  - `__ostream_type & operator<< (__streambuf_type *__sb)`
  - `__ostream_type & operator<< (const void *__p)`
  - `__ostream_type & operator<< (nullptr_t)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
  - `iostate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `__ostream_type & seekp (pos_type)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iostate __state)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`
  - `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
  - `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`
- `operator bool () const`
- `bool operator! () const`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Public Attributes

- class `__attribute__((__abi_tag__("cxx11")))` failure typedef `_los_Fmtflags` `fmtflags`

### Static Public Attributes

- static const `openmode` `__noreplace`
- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`

- static const [seekdir beg](#)
- static const [openmode binary](#)
- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iosstate eofbit](#)
- static const [iosstate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iosstate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

## Protected Types

- enum

## Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) & \_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- [\\_\\_ostream\\_type](#) & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) & \_\_rhs) noexcept
- [basic\\_ostream](#) ()
- [basic\\_ostream](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &)
- [basic\\_ostream](#) ([basic\\_ostream](#) && \_\_rhs)
- [basic\\_ostream](#) (const [basic\\_ostream](#) &)=delete
- void [init](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
- void [move](#) ([basic\\_ios](#) && \_\_rhs)
- void [move](#) ([basic\\_ios](#) & \_\_rhs)
- [basic\\_ostream](#) & [operator=](#) ([basic\\_ostream](#) && \_\_rhs)
- [basic\\_ostream](#) & [operator=](#) (const [basic\\_ostream](#) &)=delete
- void [set\\_rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
- void [swap](#) ([basic\\_ios](#) & \_\_rhs) noexcept
- void [swap](#) ([basic\\_ostream](#) & \_\_rhs)

### Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

#### 5.883.1 Detailed Description

`template<typename _CharT, typename _Traits>`  
**class** `std::basic_ostream<_CharT, _Traits>::sentry`

Performs setup work for output streams.

Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

#### 5.883.2 Member Typedef Documentation

##### `__num_get_type`

```
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type
```

These are non-standard types.

##### `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)
```

The type of an event callback function.

##### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the ios_base object.                      |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().



**iostate**

```
typedef _Ios_Iostate std::ios_base::iostate
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

**openmode**

```
typedef _Ios_Openmode std::ios_base::openmode
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

**5.883.3 Member Enumeration Documentation****event**

```
enum std::ios_base::event
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

### 5.883.4 Constructor & Destructor Documentation

#### sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::sentry (
    basic_ostream< _CharT, _Traits > & __os) [explicit]
```

The constructor performs preparatory work.

#### Parameters

|                   |                             |
|-------------------|-----------------------------|
| <code>__os</code> | The output stream to guard. |
|-------------------|-----------------------------|

If the stream state is good (`__os.good()` is true), then if the stream is tied to another output stream, `is.↵tie() -> flush()` is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

References [std::basic\\_ios<\\_CharT, \\_Traits>::bad\(\)](#), [basic\\_ostream\(\)](#), [std::ios\\_base::failbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::good\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::tie\(\)](#).

#### ~sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::~sentry () [inline]
```

Possibly flushes the stream.

If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor flushes the output stream.

References [std::ios\\_base::badbit](#), [std::uncaught\\_exception\(\)](#), and [std::ios\\_base::unitbuf](#).

#### ~basic\_ostream()

```
virtual std::basic_ostream< _CharT, _Traits >::~~basic_ostream () [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

### 5.883.5 Member Function Documentation

#### \_M\_getloc()

```
const locale & std::ios_base::_M_getloc () const [inline]
```

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

#### bad()

```
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard]
```

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

**basic\_ostream()**

```
std::basic_ostream< _CharT, _Traits >::basic_ostream (
    __streambuf_type * __sb) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Referenced by [sentry\(\)](#).

**clear()**

```
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit)
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

**copyfmt()**

```
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios & __rhs)
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

**eof()**

```
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

**exceptions() [1/2]**

```
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

**exceptions()** [2/2]

```
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except) [inline]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

**fill()** [1/2]

```
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

**fill()** [2/2]

```
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch) [inline]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags() [1/2]**

```
fmtflags std::ios_base::flags () const [inline], [nodiscard]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

**flags() [2/2]**

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**flush()**

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()
```

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

**good()**

```
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around rdstate.

**imbue()**

```
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc)
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

**init()**

```
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb) [protected]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

**iword()**

```
long & std::ios_base::iword (
    int __ix) [inline]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**narrow()**

```
char std::basic_ios< _CharT, _Traits >::narrow (
    char_type __c,
    char __default) const [inline]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool() [1/2]**

```
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator bool() [2/2]**

```
template<typename _CharT, typename _Traits>
```

```
std::basic_ostream< _CharT, _Traits >::sentry::operator bool () const [inline], [explicit]
```

Quick status checking.

**Returns**

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

**operator"!"()**

```
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator<<() [1/17]**

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(* __pf ) (__ios_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `io manip` header.

**operator<<()** [2/17]

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    __ostream_type & (* __pf) (__ostream_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<()** [3/17]

```
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    __streambuf_type * __sb)
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

**operator<<()** [4/17]

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    bool __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [5/17]

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    const void * __p) [inline]
```

Pointer arithmetic inserters.



**Parameters**

|                                    |                             |
|------------------------------------|-----------------------------|
| <code>_↵</code><br><code>_p</code> | A variable of pointer type. |
|------------------------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [6/17]**

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (  
    double __f) [inline]
```

Floating point arithmetic inserters.

**Parameters**

|                                                                                          |                                            |
|------------------------------------------------------------------------------------------|--------------------------------------------|
| <code>↵</code><br><code>_↵</code><br><code>↵</code><br><code>_↵</code><br><code>f</code> | A variable of builtin floating point type. |
|------------------------------------------------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [7/17]**

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (  
    float __f) [inline]
```

Floating point arithmetic inserters.

**Parameters**

|                                                                                          |                                            |
|------------------------------------------------------------------------------------------|--------------------------------------------|
| <code>↵</code><br><code>_↵</code><br><code>↵</code><br><code>_↵</code><br><code>f</code> | A variable of builtin floating point type. |
|------------------------------------------------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [8/17]**

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (  
    int __n)
```

Integer arithmetic inserters.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [9/17]

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    ios_base &(* __pf) (ios_base &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `io manip` header.

**operator<<()** [10/17]

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    long __n) [inline]
```

Integer arithmetic inserters.

## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    long double __f) [inline]
```

Floating point arithmetic inserters.

## Parameters

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| $f$             |                                            |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [12/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (  
    long long __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [13/17]

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (  
    short __n)
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [14/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (  
    unsigned int __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [15/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (  
    unsigned long __n) [inline]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [16/17]

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned long long __n) [inline]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [17/17]

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned short __n) [inline]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard]
```

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

**precision()** [2/2]

```
streamsize std::ios_base::precision (  
    streamsize __prec) [inline]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of precision().

**put()**

```
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (
    char_type __c)
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

**pword()**

```
void *& std::ios_base::pword (
    int __ix) [inline]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf() [1/2]**

```
basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

**rdbuf()** [2/2]

```
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb)
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

**rdstate()**

```
ios_state std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

**register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index)
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**seekp()** [1/2]

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    off_type __off,
    ios_base::seekdir __dir)
```

Changing the current write position.

## Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

## Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

**seekp()** [2/2]

```
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (  
    pos_type __pos)
```

Changing the current write position.

## Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

## Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

**setf()** [1/2]

```
fmtflags std::ios_base::setf (  
    fmtflags __fmtfl) [inline]
```

Setting new format flags.

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

## Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

**setf()** [2/2]

```
fmtflags std::ios_base::setf (  
    fmtflags __fmtfl,  
    fmtflags __mask) [inline]
```

Setting new format flags.

## Parameters

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |



**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

**setstate()**

```
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state) [inline]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

**sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellp()**

```
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

**tie() [1/2]**

```
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

**tie()** [2/2]

```
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie (
    basic_ostream<_CharT, _Traits> * __tiestr) [inline]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

**unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
char_type std::basic_ios<_CharT, _Traits>::widen (
    char __c) const [inline]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

**width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**write()**

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n)
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
static int std::ios_base::xalloc () throw ( ) [static]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `word` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `word` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `word` and `pword` arrays.

### 5.883.6 Member Data Documentation

#### adjustfield

const [fmtflags](#) std::ios\_base::adjustfield [static]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

#### app

const [openmode](#) std::ios\_base::app [static]

Seek to end before each write.

#### ate

const [openmode](#) std::ios\_base::ate [static]

Open and seek to end immediately after opening.

#### badbit

const [iostate](#) std::ios\_base::badbit [static]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

#### basefield

const [fmtflags](#) std::ios\_base::basefield [static]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

#### beg

const [seekdir](#) std::ios\_base::beg [static]

Request a seek relative to the beginning of the stream.

#### binary

const [openmode](#) std::ios\_base::binary [static]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

#### boolalpha

const [fmtflags](#) std::ios\_base::boolalpha [static]

Insert/extract `bool` in alphabetic rather than numeric format.

#### cur

const [seekdir](#) std::ios\_base::cur [static]

Request a seek relative to the current position within the sequence.

#### dec

const [fmtflags](#) std::ios\_base::dec [static]

Converts integer input or generates integer output in decimal base.

**end**

```
const seekdir std::ios_base::end [static]
```

Request a seek relative to the current end of the sequence.

**eofbit**

```
const iostate std::ios_base::eofbit [static]
```

Indicates that an input operation reached the end of an input sequence.

**failbit**

```
const iostate std::ios_base::failbit [static]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

**fixed**

```
const fmtflags std::ios_base::fixed [static]
```

Generate floating-point output in fixed-point notation.

**floatfield**

```
const fmtflags std::ios_base::floatfield [static]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

**fmtflags**

```
class __attribute__((__abi_tag__ ("cxx11"))) failure typedef _Ios_Fmtflags std::ios_base::fmtflags
```

These are thrown to indicate problems with io.

#### 27.4.2.1.1 Class `ios_base::failure`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`

- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

### goodbit

```
const ios_base::goodbit [static]
```

Indicates all is well.

### hex

```
const fmtflags std::ios_base::hex [static]
```

Converts integer input or generates integer output in hexadecimal base.

### in

```
const openmode std::ios_base::in [static]
```

Open for input. Default for `ifstream` and `fstream`.

### internal

```
const fmtflags std::ios_base::internal [static]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

### left

```
const fmtflags std::ios_base::left [static]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

### oct

```
const fmtflags std::ios_base::oct [static]
```

Converts integer input or generates integer output in octal base.

### out

```
const openmode std::ios_base::out [static]
```

Open for output. Default for `ofstream` and `fstream`.

### right

```
const fmtflags std::ios_base::right [static]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

### scientific

```
const fmtflags std::ios_base::scientific [static]
```

Generates floating-point output in scientific notation.

**showbase**

```
const fmtflags std::ios_base::showbase [static]
```

Generates a prefix indicating the numeric base of generated integer output.

**showpoint**

```
const fmtflags std::ios_base::showpoint [static]
```

Generates a decimal-point character unconditionally in generated floating-point output.

**showpos**

```
const fmtflags std::ios_base::showpos [static]
```

Generates a + sign in non-negative generated numeric output.

**skipws**

```
const fmtflags std::ios_base::skipws [static]
```

Skips leading white space before certain input operations.

**trunc**

```
const openmode std::ios_base::trunc [static]
```

Truncate an existing stream when opening. Default for ofstream.

**unitbuf**

```
const fmtflags std::ios_base::unitbuf [static]
```

Flushes output after each output operation.

**uppercase**

```
const fmtflags std::ios_base::uppercase [static]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

The documentation for this class was generated from the following files:

- [ostream.h](#)
- [ostream.tcc](#)

**5.884 \_\_gnu\_pbds::sequence\_tag Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::sequence\_tag:



#### 5.884.1 Detailed Description

Basic sequence.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.885 \_\_gnu\_parallel::sequential\_tag Struct Reference

```
#include <tags.h>
```

#### 5.885.1 Detailed Description

Forces sequential execution at compile time.

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.886 std::\_\_debug::set< \_Key, \_Compare, \_Allocator > Class Template Reference

```
#include <set.h>
```

Inheritance diagram for `std::__debug::set< _Key, _Compare, _Allocator >`:





## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_const_iterator, set >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- using **insert\_return\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_iterator, set >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- using **node\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **set** (`_Base_ref __x`)
- template<typename `_InputIterator`>  
  **set** (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator()`)
- template<typename `_InputIterator`>  
  **set** (`_InputIterator __first, _InputIterator __last, const allocator_type &__a`)
- **set** (`const _Compare &__comp, const _Allocator &__a=_Allocator()`)
- **set** (`const allocator_type &__a`)
- **set** (`const set &`)=default
- **set** (`const set &__x, const __type_identity_t< allocator_type > &__a`)
- **set** (`(initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())`)
- **set** (`(initializer_list< value_type > __l, const allocator_type &__a`)
- **set** (`(set &&)=default`)
- **set** (`(set &&__x, const __type_identity_t< allocator_type > &__a) noexcept(noexcept(_Base(std::move(__x), __a)))`)
- `__attribute__((abi_tag("cxx11")))` **iterator** `erase(const_iterator __first`
- `__attribute__((abi_tag("cxx11")))` **iterator** `erase(const_iterator __position)`
- `const _Base & _M_base ()` const noexcept
- `_Base & _M_base ()` noexcept
- `const_iterator` **begin** () const noexcept
- `iterator` **begin** () noexcept
- `const_iterator` **cbegin** () const noexcept
- `const_iterator` **cend** () const noexcept
- void **clear** () noexcept
- `const_reverse_iterator` **crbegin** () const noexcept
- `const_reverse_iterator` **crend** () const noexcept
- template<typename... `_Args`>  
  `std::pair< iterator, bool >` **emplace** (`_Args &&... __args`)

- `template<typename... _Args>`  
`iterator` **emplace\_hint** (`const_iterator` \_\_pos, `_Args` &&... \_\_args)
- `const_iterator` **end** () `const noexcept`
- `iterator` **end** () `noexcept`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< iterator, iterator >` **equal\_range** (`const _Kt` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< const_iterator, const_iterator >` **equal\_range** (`const _Kt` &\_\_x) `const`
- `std::pair< iterator, iterator >` **equal\_range** (`const key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (`const key_type` &\_\_x) `const`
- `_Base_iterator` **erase** (`_Base_const_iterator` \_\_position)
- `size_type` **erase** (`const key_type` &\_\_x)
- `node_type` **extract** (`const key_type` &\_\_key)
- `node_type` **extract** (`const_iterator` \_\_position)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **find** (`const _Kt` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **find** (`const _Kt` &\_\_x) `const`
- `iterator` **find** (`const key_type` &\_\_x)
- `const_iterator` **find** (`const key_type` &\_\_x) `const`
- `for` (`_Base_const_iterator` \_\_victim=\_\_first.base(); \_\_victim != \_\_last.base(); ++ \_\_victim)
- `template<typename _InputIterator>`  
`void` **insert** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `std::pair< iterator, bool >` **insert** (`const value_type` &\_\_x)
- `iterator` **insert** (`const_iterator` \_\_hint, `node_type` &&\_\_nh)
- `iterator` **insert** (`const_iterator` \_\_position, `const value_type` &\_\_x)
- `iterator` **insert** (`const_iterator` \_\_position, `value_type` &&\_\_x)
- `void` **insert** (`initializer_list< value_type >` \_\_l)
- `insert_return_type` **insert** (`node_type` &&\_\_nh)
- `std::pair< iterator, bool >` **insert** (`value_type` &&\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **lower\_bound** (`const _Kt` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **lower\_bound** (`const _Kt` &\_\_x) `const`
- `iterator` **lower\_bound** (`const key_type` &\_\_x)
- `const_iterator` **lower\_bound** (`const key_type` &\_\_x) `const`
- `set` & **operator=** (`const set` &)=default
- `set` & **operator=** (`initializer_list< value_type >` \_\_l)
- `set` & **operator=** (`set` &&)=default
- `const_reverse_iterator` **rbegin** () `const noexcept`
- `reverse_iterator` **rbegin** () `noexcept`
- `const_reverse_iterator` **rend** () `const noexcept`
- `reverse_iterator` **rend** () `noexcept`
- `void` **swap** (`set` &\_\_x) `noexcept(/*conditional */)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **upper\_bound** (`const _Kt` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **upper\_bound** (`const _Kt` &\_\_x) `const`
- `iterator` **upper\_bound** (`const key_type` &\_\_x)
- `const_iterator` **upper\_bound** (`const key_type` &\_\_x) `const`
- `while` (false)

### Public Attributes

- [const\\_iterator](#) `__last`
- `do`
- `return`

### Protected Member Functions

- `constexpr void _M_swap (const _Safe_container &__x) const noexcept`

### Friends

- `template<typename _ItT, typename _SeqT, typename _CatT>`  
`class ::__gnu_debug::_Safe_iterator`

### 5.886.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>`

`class std::__debug::set< _Key, _Compare, _Allocator >`

Class `std::set` with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [set.h](#)

### 5.887 `std::set< _Key, _Compare, _Alloc >` Class Template Reference

```
#include <set>
```

### Public Types

- `typedef _Key` [key\\_type](#)
- `typedef _Key` [value\\_type](#)
- `typedef _Compare` [key\\_compare](#)
- `typedef _Compare` [value\\_compare](#)
- `typedef _Alloc` [allocator\\_type](#)
  
- `typedef _Alloc_traits::pointer` [pointer](#)
- `typedef _Alloc_traits::const_pointer` [const\\_pointer](#)
- `typedef _Alloc_traits::reference` [reference](#)
- `typedef _Alloc_traits::const_reference` [const\\_reference](#)
- `typedef _Rep_type::const_iterator` [iterator](#)
- `typedef _Rep_type::const_iterator` [const\\_iterator](#)
- `typedef _Rep_type::const_reverse_iterator` [reverse\\_iterator](#)
- `typedef _Rep_type::const_reverse_iterator` [const\\_reverse\\_iterator](#)
- `typedef _Rep_type::size_type` [size\\_type](#)
- `typedef _Rep_type::difference_type` [difference\\_type](#)

## Public Member Functions

- [set](#) ()=default
- [template<typename \\_InputIterator>](#)  
[set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [template<typename \\_InputIterator>](#)  
[set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [template<typename \\_InputIterator>](#)  
[set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const [allocator\\_type](#) &\_\_a)
- [set](#) (const \_Compare &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [set](#) (const [allocator\\_type](#) &\_\_a)
- [set](#) (const [set](#) &)=default
- [set](#) (const [set](#) &\_\_x, const \_\_type\_identity\_t< [allocator\\_type](#) > &\_\_a)
- [set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const \_Compare &\_\_comp=\_Compare(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- [set](#) ([set](#) &&)=default
- [set](#) ([set](#) &&\_\_x, const \_\_type\_identity\_t< [allocator\\_type](#) > &\_\_a) noexcept([is\\_nothrow\\_copy\\_constructible](#)< \_\_Compare >::value &&\_Alloc\_traits::\_S\_always\_equal())
- [~set](#) ()=default
- [iterator begin](#) () const noexcept
- [iterator cbegin](#) () const noexcept
- [iterator cend](#) () const noexcept
- [void clear](#) () noexcept
- [reverse\\_iterator crbegin](#) () const noexcept
- [reverse\\_iterator crend](#) () const noexcept
- [template<typename... \\_Args>](#)  
[std::pair](#)< [iterator](#), bool > [emplace](#) (\_Args &&... \_\_args)
- [template<typename... \\_Args>](#)  
[iterator emplace\\_hint](#) (const [iterator](#) \_\_pos, \_Args &&... \_\_args)
- [bool empty](#) () const noexcept
- [iterator end](#) () const noexcept
- [size\\_type erase](#) (const [key\\_type](#) &\_\_x)
- [\\_GLIBCXX\\_ABI\\_TAG\\_CXX11 iterator erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [\\_GLIBCXX\\_ABI\\_TAG\\_CXX11 iterator erase](#) (const [iterator](#) \_\_position)
- [allocator\\_type get\\_allocator](#) () const noexcept
- [template<typename \\_InputIterator>](#)  
[void insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [std::pair](#)< [iterator](#), bool > [insert](#) (const [value\\_type](#) &\_\_x)
- [iterator insert](#) (const [iterator](#) \_\_position, const [value\\_type](#) &\_\_x)
- [iterator insert](#) (const [iterator](#) \_\_position, [value\\_type](#) &&\_\_x)
- [void insert](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [std::pair](#)< [iterator](#), bool > [insert](#) ([value\\_type](#) &&\_\_x)
- [key\\_compare key\\_comp](#) () const
- [size\\_type max\\_size](#) () const noexcept
- [set & operator=](#) (const [set](#) &)=default
- [set & operator=](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [set & operator=](#) ([set](#) &&)=default
- [reverse\\_iterator rbegin](#) () const noexcept
- [reverse\\_iterator rend](#) () const noexcept
- [size\\_type size](#) () const noexcept

- `void swap (set &__x) noexcept(/*conditional */)`
- `value_compare value_comp () const`
  
- `size_type count (const key_type &__x) const`
- `template<typename _Kt>`  
`auto count (const _Kt &__x) const -> decltype(_M_t._M_count_tr(__x))`
  
- `bool contains (const key_type &__x) const`
- `template<typename _Kt>`  
`auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)`
  
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `template<typename _Kt>`  
`auto find (const _Kt &__x) -> decltype(iterator{_M_t._M_find_tr(__x)})`
- `template<typename _Kt>`  
`auto find (const _Kt &__x) const -> decltype(const_iterator{_M_t._M_find_tr(__x)})`
  
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `template<typename _Kt>`  
`auto lower_bound (const _Kt &__x) -> decltype(iterator{_M_t._M_lower_bound_tr(__x)})`
- `template<typename _Kt>`  
`auto lower_bound (const _Kt &__x) const -> decltype(const_iterator{_M_t._M_lower_bound_tr(__x)})`
  
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `template<typename _Kt>`  
`auto upper_bound (const _Kt &__x) -> decltype(iterator{_M_t._M_upper_bound_tr(__x)})`
- `template<typename _Kt>`  
`auto upper_bound (const _Kt &__x) const -> decltype(const_iterator{_M_t._M_upper_bound_tr(__x)})`
  
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `template<typename _Kt>`  
`auto equal_range (const _Kt &__x) -> decltype(pair< iterator, iterator >{_M_t._M_equal_range_tr(__x)})`
- `template<typename _Kt>`  
`auto equal_range (const _Kt &__x) const -> decltype(pair< const_iterator, const_iterator >{_M_t._M_equal_range_tr(__x)})`

## Friends

- `template<typename _K1, typename _C1, typename _A1>`  
`bool operator< (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`
- `template<typename _K1, typename _C1, typename _A1>`  
`bool operator== (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`

### 5.887.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::set<_Key, _Compare, _Alloc>
```

A standard container made up of unique keys, which can be retrieved in logarithmic time.

Since

C++98

#### Template Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                         |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> . |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .             |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (\*\_unique versus \*\_equal, same as the standard).

### 5.887.2 Member Typedef Documentation

#### allocator\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typedef _Alloc std::set<_Key, _Compare, _Alloc>::allocator_type
```

Public typedefs.

#### const\_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typedef _Rep_type::const_iterator std::set<_Key, _Compare, _Alloc>::const_iterator
```

Iterator-related typedefs.

#### const\_pointer

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typedef _Alloc_traits::const_pointer std::set<_Key, _Compare, _Alloc>::const_pointer
```

Iterator-related typedefs.

#### const\_reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typedef _Alloc_traits::const_reference std::set<_Key, _Compare, _Alloc>::const_reference
```

Iterator-related typedefs.

**const\_reverse\_iterator**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::const_reverse_↵
iterator
```

Iterator-related typedefs.

**difference\_type**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::difference_type std::set< _Key, _Compare, _Alloc >::difference_type
```

Iterator-related typedefs.

**iterator**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::iterator
```

Iterator-related typedefs.

**key\_compare**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Compare std::set< _Key, _Compare, _Alloc >::key_compare
```

Public typedefs.

**key\_type**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::key_type
```

Public typedefs.

**pointer**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::pointer std::set< _Key, _Compare, _Alloc >::pointer
```

Iterator-related typedefs.

**reference**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::reference std::set< _Key, _Compare, _Alloc >::reference
```

Iterator-related typedefs.

**reverse\_iterator**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::reverse_iterator
```

Iterator-related typedefs.

**size\_type**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typename _Rep_type::size_type std::set<_Key, _Compare, _Alloc>::size_type
```

Iterator-related typedefs.

**value\_compare**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typename _Compare std::set<_Key, _Compare, _Alloc>::value_compare
```

Public typedefs.

**value\_type**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typename _Key std::set<_Key, _Compare, _Alloc>::value_type
```

Public typedefs.

**5.887.3 Constructor & Destructor Documentation****set()** [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set () [default]
```

Default constructor creates no elements.

**set()** [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
    const _Compare & __comp,
    const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a set with no elements.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | Comparator to use.   |
| <code>__a</code>    | An allocator object. |

**set()** [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator>
std::set<_Key, _Compare, _Alloc>::set (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

Builds a set from a range.



**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a set consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

**set()** [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator>
std::set<_Key, _Compare, _Alloc>::set (
    _InputIterator __first,
    _InputIterator __last,
    const _Compare & __comp,
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a set from a range.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a set consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

**set()** [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
    const set<_Key, _Compare, _Alloc> & ) [default]
```

Set copy constructor.

Whether the allocator is copied depends on the allocator traits.

**set()** [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
    set<_Key, _Compare, _Alloc> && ) [default]
```

Set move constructor

The newly-created set contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, set.

**set()** [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::set<_Key, _Compare, _Alloc >::set (
    initializer_list< value_type > __l,
    const _Compare & __comp = _Compare(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds a set from an initializer\_list.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__l</code>    | An initializer_list.  |
| <code>__comp</code> | A comparison functor. |
| <code>__a</code>    | An allocator object.  |

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

#### set() [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc >::set (
    const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

#### set() [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc >::set (
    const set<_Key, _Compare, _Alloc > & __x,
    const __type_identity_t< allocator_type > & __a) [inline]
```

Allocator-extended copy constructor.

#### set() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc >::set (
    set<_Key, _Compare, _Alloc > && __x,
    const __type_identity_t< allocator_type > & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

#### set() [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc >::set (
    initializer_list< value_type > __l,
    const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

#### set() [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _InputIterator>
std::set< _Key, _Compare, _Alloc >::set (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

### ~set()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::~set () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 5.887.4 Member Function Documentation

### begin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

### cbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

### cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

### clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::set< _Key, _Compare, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### contains() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::contains (
    const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true) [inline]
```

Finds whether an element with the given key exists.

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <code>_↔</code>  | Key of elements to be located. |
| <code>__x</code> |                                |

#### Returns

True if there is an element with the specified key.

#### contains() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
bool std::set<_Key, _Compare, _Alloc >::contains (
    const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <code>_↔</code>  | Key of elements to be located. |
| <code>__x</code> |                                |

#### Returns

True if there is an element with the specified key.

#### count() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
template<typename _Kt>
auto std::set<_Key, _Compare, _Alloc >::count (
    const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x)) [inline]
```

Finds the number of elements.

#### Parameters

|                  |                     |
|------------------|---------------------|
| <code>_↔</code>  | Element to located. |
| <code>__x</code> |                     |

#### Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

#### count() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
size_type std::set<_Key, _Compare, _Alloc >::count (
    const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                 |                     |
|-----------------|---------------------|
| <code>_↔</code> | Element to located. |
| <code>_X</code> |                     |

**Returns**

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

**crbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔  
_Key>>
```

```
reverse_iterator std::set< _Key, _Compare, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

**crend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔  
_Key>>
```

```
reverse_iterator std::set< _Key, _Compare, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

**emplace()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔  
_Key>>
```

```
template<typename... _Args>
```

```
std::pair< iterator, bool > std::set< _Key, _Compare, _Alloc >::emplace (   
_Args &&... __args) [inline]
```

Attempts to build and insert an element into the set.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

**emplace\_hint()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔  
_Key>>
```

```
template<typename... _Args>
```

```
iterator std::set< _Key, _Compare, _Alloc >::emplace_hint (
```

```
const_iterator __pos,
_Args &&... __args) [inline]
```

Attempts to insert an element into the set.

#### Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

#### Returns

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

#### `empty()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
bool std::set<_Key, _Compare, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the set is empty.

#### `end()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
iterator std::set<_Key, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

#### `equal_range()` [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _Kt>
```

```
auto std::set<_Key, _Compare, _Alloc >::equal_range (
    const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
```

```
[inline]
```

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**equal\_range() [2/4]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
template<typename _Kt>  
auto std::set< _Key, _Compare, _Alloc >::equal_range (  
    const _Kt & __x) const -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_↵  
tr(__x)))    [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                |                    |
|----------------|--------------------|
| <code>↵</code> | Key to be located. |
| <code>X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**equal\_range() [3/4]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
std::pair< iterator, iterator > std::set< _Key, _Compare, _Alloc >::equal_range (  
    const key_type & __x)    [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                |                    |
|----------------|--------------------|
| <code>↵</code> | Key to be located. |
| <code>X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**equal\_range()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair< const_iterator, const_iterator > std::set< _Key, _Compare, _Alloc >::equal_range (
    const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**erase()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::set< _Key, _Compare, _Alloc >::erase (
    const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from a set.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
|----------------------|-----------------------------------------------------------|



|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <code>__last</code> | Iterator pointing to the end of the range to be erased. |
|---------------------|---------------------------------------------------------|

#### Returns

The iterator `__last`.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### **erase()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase (
    const_iterator __position) [inline]
```

Erases an element from a set.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

#### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### **find()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::find (
    const _Kt & __x) -> decltype(iterator{_M_t._M_find_tr(__x)}) [inline]
```

Tries to locate an element in a set.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

#### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**find()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::set<_Key, _Compare, _Alloc>::find (
    const _Kt & __x) const -> decltype(const_iterator{M_t.M_find_tr(__x)})    [inline]
```

Tries to locate an element in a set.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc>::find (
    const key_type & __x)    [inline]
```

Tries to locate an element in a set.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set<_Key, _Compare, _Alloc>::find (
    const key_type & __x) const    [inline]
```

Tries to locate an element in a set.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
allocator_type std::set< _Key, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]
```

Returns the allocator object with which the set was constructed.

**insert() [1/4]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _InputIterator>
```

```
void std::set< _Key, _Compare, _Alloc >::insert (
```

```
    _InputIterator __first,
```

```
    _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert() [2/4]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::pair< iterator, bool > std::set< _Key, _Compare, _Alloc >::insert (
    const value_type & __x) [inline]
```

Attempts to insert an element into the set.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Referenced by `std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::insert()`.

**insert() [3/4]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
iterator std::set< _Key, _Compare, _Alloc >::insert (
```

```
    const_iterator __position,
```

```
    const value_type & __x) [inline]
```

Attempts to insert an element into the set.

## Parameters

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>        | Element to be inserted.                                                       |

## Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::set<_Key, _Compare, _Alloc>::insert (
    initializer_list<value_type> & __l) [inline]
```

Attempts to insert a list of elements into the set.

## Parameters

|                |                                                                                    |
|----------------|------------------------------------------------------------------------------------|
| <code>↵</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
| <code>↵</code> |                                                                                    |
| <code>↵</code> |                                                                                    |
| <code>↵</code> |                                                                                    |
| <code>/</code> |                                                                                    |

Complexity similar to that of the range constructor.

**key\_comp()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
key_compare std::set<_Key, _Compare, _Alloc>::key_comp () const [inline]
```

Returns the comparison object with which the set was constructed.

**lower\_bound()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::set<_Key, _Compare, _Alloc>::lower_bound (
    const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>↵</code>   | Key to be located. |
| <code>__x</code> |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::lower_bound (
    const _Kt & __x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                                   |                    |
|-----------------------------------|--------------------|
| <code>↵</code><br><code>_X</code> | Key to be located. |
|-----------------------------------|--------------------|

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
    const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                                   |                    |
|-----------------------------------|--------------------|
| <code>↵</code><br><code>_X</code> | Key to be located. |
|-----------------------------------|--------------------|

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set<_Key, _Compare, _Alloc >::lower_bound (
    const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**max\_size()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
size_type std::set< _Key, _Compare, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the set.

**operator=()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
set & std::set< _Key, _Compare, _Alloc >::operator= (
    const set< _Key, _Compare, _Alloc > & ) [default]
```

Set assignment operator.

Whether the allocator is copied depends on the allocator traits.

**operator=()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
set & std::set< _Key, _Compare, _Alloc >::operator= (
    initializer_list< value_type > __l) [inline]
```

Set list assignment operator.

**Parameters**

|                 |                      |
|-----------------|----------------------|
| <code>↵</code>  | An initializer_list. |
| <code>_↵</code> |                      |
| <code>↵</code>  |                      |
| <code>_↵</code> |                      |
| <code>/</code>  |                      |

This function fills a set with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
set & std::set< _Key, _Compare, _Alloc >::operator= (
    set< _Key, _Compare, _Alloc > && ) [default]
```

Move assignment operator.

**rbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
reverse_iterator std::set<_Key, _Compare, _Alloc>::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

**rend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
reverse_iterator std::set<_Key, _Compare, _Alloc>::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

**size()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
size_type std::set<_Key, _Compare, _Alloc>::size () const [inline], [noexcept]
```

Returns the size of the set.

**swap()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
void std::set<_Key, _Compare, _Alloc>::swap (
    set<_Key, _Compare, _Alloc> & __x) [inline], [noexcept]
```

Swaps data with another set.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__x</code> | A set of the same element and allocator types. |
|------------------|------------------------------------------------|

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Referenced by [std::swap\(\)](#).

**upper\_bound()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _Kt>
```

```
auto std::set<_Key, _Compare, _Alloc>::upper_bound (
    const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x))) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|



**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::upper_bound (
    const _Kt & __x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::upper_bound (
    const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::upper_bound (
    const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to the first element greater than key, or end().

**value\_comp()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
value_compare std::set<_Key, _Compare, _Alloc>::value_comp () const [inline]
```

Returns the comparison object with which the set was constructed.

The documentation for this class was generated from the following files:

- [stl\\_multiset.h](#)
- [stl\\_set.h](#)

**5.888 `std::shared_future<_Res>` Class Template Reference**

```
#include <future>
```

Inheritance diagram for `std::shared_future<_Res>`:

**Public Member Functions**

- [shared\\_future](#) (const [shared\\_future](#) &\_\_sf) noexcept
- [shared\\_future](#) (future<\_Res> &&\_\_uf) noexcept
- [shared\\_future](#) (shared\_future &&\_\_sf) noexcept
- const \_Res & [get](#) () const
- [shared\\_future](#) & **operator=** (const [shared\\_future](#) &\_\_sf) noexcept
- [shared\\_future](#) & **operator=** (shared\_future &&\_\_sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- template<typename \_Rep, typename \_Period>  
[future\\_status](#) **wait\_for** (const chrono::duration<\_Rep, \_Period> &\_\_rel) const
- template<typename \_Clock, typename \_Duration>  
[future\\_status](#) **wait\_until** (const chrono::time\_point<\_Clock, \_Duration> &\_\_abs) const

## Protected Types

- typedef `__future_base::Result<_Res>` & `__result_type`
- typedef `shared_ptr<_State_base>` `__state_type`

## Protected Member Functions

- `__result_type M_get_result()` const
- void `M_swap (__basic_future &__that)` noexcept

### 5.888.1 Detailed Description

```
template<typename _Res>
class std::shared_future<_Res>
```

Primary template for `shared_future`.

### 5.888.2 Constructor & Destructor Documentation

#### `shared_future()` [1/3]

```
template<typename _Res>
std::shared_future<_Res>::shared_future (
    const shared_future<_Res> & __sf) [inline], [noexcept]
```

Copy constructor.

#### `shared_future()` [2/3]

```
template<typename _Res>
std::shared_future<_Res>::shared_future (
    future<_Res> && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

#### `shared_future()` [3/3]

```
template<typename _Res>
std::shared_future<_Res>::shared_future (
    shared_future<_Res> && __sf) [inline], [noexcept]
```

Construct from a `shared_future` rvalue.

### 5.888.3 Member Function Documentation

#### `M_get_result()`

```
template<typename _Res>
__result_type std::__basic_future<_Res>::M_get_result () const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

#### `get()`

```
template<typename _Res>
const _Res & std::shared_future<_Res>::get () const [inline]
```

Retrieving the value.

The documentation for this class was generated from the following file:

- `future`

## 5.889 std::shared\_future<\_Res & > Class Template Reference

```
#include <future>
```

Inheritance diagram for std::shared\_future<\_Res & >:



### Public Member Functions

- `shared_future` (const `shared_future` & \_\_sf)
- `shared_future` (const `shared_future` & \_\_sf) noexcept
- `shared_future` (`future`<\_Res & > && \_\_uf) noexcept
- `shared_future` (`future`<\_Res > && \_\_uf) noexcept
- `shared_future` (`shared_future` && \_\_sf) noexcept
- `shared_future` (`shared_future` && \_\_sf) noexcept
- const \_Res & `get` () const
- \_Res & `get` () const
- `shared_future` & `operator=` (const `shared_future` & \_\_sf)
- `shared_future` & `operator=` (const `shared_future` & \_\_sf) noexcept
- `shared_future` & `operator=` (`shared_future` && \_\_sf) noexcept
- `shared_future` & `operator=` (`shared_future` && \_\_sf) noexcept
- bool `valid` () const noexcept
- void `wait` () const
- `future_status` `wait_for` (const `chrono::duration`<\_Rep, \_Period > & \_\_rel) const
- `future_status` `wait_until` (const `chrono::time_point`<\_Clock, \_Duration > & \_\_abs) const

### Protected Types

- typedef \_\_future\_base::\_Result<\_Res > & \_\_result\_type
- typedef `shared_ptr`<\_State\_base > \_\_state\_type

### Protected Member Functions

- \_\_result\_type `_M_get_result` () const
- void `_M_swap` (\_\_basic\_future & \_\_that) noexcept

### 5.889.1 Detailed Description

```
template<typename _Res>
class std::shared_future<_Res & >
```

Partial specialization for `shared_future<R&>`

### 5.889.2 Constructor & Destructor Documentation

#### `shared_future()` [1/6]

```
template<typename _Res>
std::shared_future<_Res & >::shared_future (
    const shared_future<_Res & > & __sf) [inline]
```

Copy constructor.

#### `shared_future()` [2/6]

```
template<typename _Res>
std::shared_future<_Res & >::shared_future (
    future<_Res & > && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

#### `shared_future()` [3/6]

```
template<typename _Res>
std::shared_future<_Res & >::shared_future (
    shared_future<_Res & > && __sf) [inline], [noexcept]
```

Construct from a `shared_future` rvalue.

#### `shared_future()` [4/6]

```
std::shared_future<_Res >::shared_future (
    const shared_future<_Res & > & __sf) [inline], [noexcept]
```

Copy constructor.

#### `shared_future()` [5/6]

```
std::shared_future<_Res >::shared_future (
    future<_Res > && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

#### `shared_future()` [6/6]

```
std::shared_future<_Res >::shared_future (
    shared_future<_Res & > && __sf) [inline], [noexcept]
```

Construct from a `shared_future` rvalue.

### 5.889.3 Member Function Documentation

#### `_M_get_result()`

```
__result_type std::__basic_future<_Res >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

**get()** [1/2]

```
const _Res & std::shared_future< _Res >::get () const [inline]
```

Retrieving the value.

**get()** [2/2]

```
template<typename _Res>
_Res & std::shared_future< _Res & >::get () const [inline]
```

Retrieving the value.

The documentation for this class was generated from the following file:

- [future](#)

**5.890 `std::shared_future< void >` Class Reference**

```
#include <future>
```

Inheritance diagram for `std::shared_future< void >`:

**Public Member Functions**

- `shared_future` (const `shared_future` &\_\_sf)
- `shared_future` (const `shared_future` &\_\_sf) noexcept
- `shared_future` (`future`< void > &&\_\_uf) noexcept
- `shared_future` (`future`< void > &&\_\_uf) noexcept
- `shared_future` (`shared_future` &&\_\_sf) noexcept
- `shared_future` (`shared_future` &&\_\_sf) noexcept
- const void & `get` () const
- void `get` () const
- `shared_future` & `operator=` (const `shared_future` &\_\_sf)
- `shared_future` & `operator=` (const `shared_future` &\_\_sf) noexcept
- `shared_future` & `operator=` (`shared_future` &&\_\_sf) noexcept
- `shared_future` & `operator=` (`shared_future` &&\_\_sf) noexcept
- bool `valid` () const noexcept
- void `wait` () const
- `future_status` `wait_for` (const `chrono::duration`< \_Rep, \_Period > &\_\_rel) const
- `future_status` `wait_until` (const `chrono::time_point`< \_Clock, \_Duration > &\_\_abs) const

## Protected Types

- typedef `__future_base::Result< void > & __result_type`
- typedef `shared_ptr< _State_base > __state_type`

## Protected Member Functions

- `__result_type _M_get_result () const`
- `void _M_swap (__basic_future &__that) noexcept`

### 5.890.1 Detailed Description

Explicit specialization for `shared_future<void>`

### 5.890.2 Constructor & Destructor Documentation

#### `shared_future()` [1/6]

```
std::shared_future< void >::shared_future (  
    const shared_future< void > & __sf) [inline]
```

Copy constructor.

#### `shared_future()` [2/6]

```
std::shared_future< void >::shared_future (  
    future< void > && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

#### `shared_future()` [3/6]

```
std::shared_future< void >::shared_future (  
    shared_future< void > && __sf) [inline], [noexcept]
```

Construct from a `shared_future` rvalue.

#### `shared_future()` [4/6]

```
std::shared_future< void >::shared_future (  
    const shared_future< void > & __sf) [inline], [noexcept]
```

Copy constructor.

#### `shared_future()` [5/6]

```
std::shared_future< void >::shared_future (  
    future< void > && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

#### `shared_future()` [6/6]

```
std::shared_future< void >::shared_future (  
    shared_future< void > && __sf) [inline], [noexcept]
```

Construct from a `shared_future` rvalue.

### 5.890.3 Member Function Documentation

#### `_M_get_result()`

```
__result_type std::__basic_future< void >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

#### `get()`

```
const void & std::shared_future< void >::get () const [inline]
```

Retrieving the value.

The documentation for this class was generated from the following file:

- [future](#)

## 5.891 `std::shared_lock<_Mutex>` Class Template Reference

```
#include <shared_mutex>
```

### Public Types

- typedef `_Mutex` **mutex\_type**

### Public Member Functions

- **shared\_lock** (mutex\_type &\_\_m)
- **shared\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#))
- template<typename \_Rep, typename \_Period>  
**shared\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rel\_time)
- template<typename \_Clock, typename \_Duration>  
**shared\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs\_time)
- **shared\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#)) noexcept
- **shared\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **shared\_lock** ([shared\\_lock](#) &&\_\_sl) noexcept
- **shared\_lock** ([shared\\_lock](#) const &)=delete
- void **lock** ()
- mutex\_type \* **mutex** () const noexcept
- **operator bool** () const noexcept
- [shared\\_lock](#) & **operator=** ([shared\\_lock](#) &&\_\_sl) noexcept
- [shared\\_lock](#) & **operator=** ([shared\\_lock](#) const &)=delete
- bool **owns\_lock** () const noexcept
- mutex\_type \* **release** () noexcept
- void **swap** ([shared\\_lock](#) &\_\_u) noexcept
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period>  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rel\_time)
- template<typename \_Clock, typename \_Duration>  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs\_time)
- void **unlock** ()



### 5.891.1 Detailed Description

```
template<typename _Mutex>
class std::shared_lock< _Mutex >
```

shared\_lock

The documentation for this class was generated from the following file:

- [shared\\_mutex](#)

## 5.892 std::shared\_ptr< \_Tp > Class Template Reference

```
#include <memory>
```

### Public Types

- using [element\\_type](#)
- using [weak\\_type](#)

### Public Member Functions

- constexpr [shared\\_ptr](#) () noexcept
- template<typename \_Yp, typename = \_Constructible<\_Yp\*>>  
[shared\\_ptr](#) (\_Yp \*\_\_p)
- template<typename \_Yp, typename \_Deleter, typename = \_Constructible<\_Yp\*, \_Deleter>>  
[shared\\_ptr](#) (\_Yp \*\_\_p, \_Deleter \_\_d)
- template<typename \_Yp, typename \_Deleter, typename \_Alloc, typename = \_Constructible<\_Yp\*, \_Deleter, \_Alloc>>  
[shared\\_ptr](#) (\_Yp \*\_\_p, \_Deleter \_\_d, \_Alloc \_\_a)
- template<typename \_Yp, typename = \_Constructible<auto\_ptr<\_Yp>>>  
[shared\\_ptr](#) (auto\_ptr<\_Yp> &&\_\_r)
- [shared\\_ptr](#) (const [shared\\_ptr](#) &) noexcept=default
- template<typename \_Yp, typename = \_Constructible<const shared\_ptr<\_Yp>&>>  
[shared\\_ptr](#) (const [shared\\_ptr](#)<\_Yp> &&\_\_r) noexcept
- template<typename \_Yp>  
[shared\\_ptr](#) (const [shared\\_ptr](#)<\_Yp> &&\_\_r, [element\\_type](#) \*\_\_p) noexcept
- template<typename \_Yp, typename = \_Constructible<const weak\_ptr<\_Yp>&>>  
[shared\\_ptr](#) (const [weak\\_ptr](#)<\_Yp> &&\_\_r)
- template<typename \_Deleter>  
[shared\\_ptr](#) (nullptr\_t \_\_p, \_Deleter \_\_d)
- template<typename \_Deleter, typename \_Alloc>  
[shared\\_ptr](#) (nullptr\_t \_\_p, \_Deleter \_\_d, \_Alloc \_\_a)
- constexpr [shared\\_ptr](#) (nullptr\_t) noexcept
- [shared\\_ptr](#) ([shared\\_ptr](#) &&\_\_r) noexcept
- template<typename \_Yp, typename = \_Constructible<shared\_ptr<\_Yp>>>  
[shared\\_ptr](#) ([shared\\_ptr](#)<\_Yp> &&\_\_r) noexcept
- template<typename \_Yp>  
[shared\\_ptr](#) ([shared\\_ptr](#)<\_Yp> &&\_\_r, [element\\_type](#) \*\_\_p) noexcept
- template<typename \_Tp1, typename>  
[shared\\_ptr](#) (std::auto\_ptr<\_Tp1> &&\_\_r)
- template<typename \_Yp, typename \_Del, typename = \_Constructible<unique\_ptr<\_Yp, \_Del>>>  
[shared\\_ptr](#) (unique\_ptr<\_Yp, \_Del> &&\_\_r)
- [element\\_type](#) \* [get](#) () const noexcept
- [operator bool](#) () const noexcept
- [element\\_type](#) & [operator\\*](#) () const noexcept

- `element_type * operator-> ()` const noexcept
- `template<typename _Yp>`  
`_Assignable< auto_ptr< _Yp > > operator= (auto_ptr< _Yp > &&__r)`
- `shared_ptr & operator= (const shared_ptr &)` noexcept=default
- `template<typename _Yp>`  
`_Assignable< const shared_ptr< _Yp > > operator= (const shared_ptr< _Yp > &&__r)` noexcept
- `shared_ptr & operator= (shared_ptr &&__r)` noexcept
- `template<class _Yp>`  
`_Assignable< shared_ptr< _Yp > > operator= (shared_ptr< _Yp > &&__r)` noexcept
- `template<typename _Yp, typename _Del>`  
`_Assignable< unique_ptr< _Yp, _Del > > operator= (unique_ptr< _Yp, _Del > &&__r)`
- `void reset ()` noexcept
- `template<typename _Yp>`  
`_SafeConv< _Yp > reset (_Yp *__p)`
- `template<typename _Yp, typename _Deleter>`  
`_SafeConv< _Yp > reset (_Yp *__p, _Deleter __d)`
- `template<typename _Yp, typename _Deleter, typename _Alloc>`  
`_SafeConv< _Yp > reset (_Yp *__p, _Deleter __d, _Alloc __a)`
- `void swap (__shared_ptr< _Tp, _Lp > &__other)` noexcept
- `bool unique ()` const noexcept
- `long use_count ()` const noexcept
- `template<typename _Tp1>`  
`bool owner_before (__shared_ptr< _Tp1, _Lp > const &__rhs)` const noexcept
- `template<typename _Tp1>`  
`bool owner_before (__weak_ptr< _Tp1, _Lp > const &__rhs)` const noexcept

## Friends

- `template<typename _Yp, typename _Alloc, typename... _Args>`  
`shared_ptr< _NonArray< _Yp > > allocate_shared (const _Alloc &, _Args &&...)`
- `template<typename _Yp, typename... _Args>`  
`shared_ptr< _NonArray< _Yp > > make_shared (_Args &&...)`
- `class weak_ptr< _Tp >`

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _Del, typename _Tp>`  
`_Del * get_deleter (const shared_ptr< _Tp > &__p)` noexcept
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename _Up>`  
`bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b)` noexcept
- `template<typename _Tp>`  
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t)` noexcept
- `template<typename _Tp>`  
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)` noexcept
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r)` noexcept

- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > static_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > const_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > dynamic_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr< _NonArray< _Tp > > allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, typename... _Args>`  
`shared_ptr< _NonArray< _Tp > > make_shared (_Args &&... __args)`

### 5.892.1 Detailed Description

`template<typename _Tp>`  
**class** `std::shared_ptr< _Tp >`

A smart pointer with reference-counted copy semantics.

Since

C++11

A `shared_ptr` object is either empty or *owns* a pointer passed to the constructor. Copies of a `shared_ptr` share ownership of the same pointer. When the last `shared_ptr` that owns the pointer is destroyed or reset, the owned pointer is freed (either by `delete` or by invoking a custom deleter that was passed to the constructor).

A `shared_ptr` also stores another pointer, which is usually (but not always) the same pointer as it owns. The stored pointer can be retrieved by calling the `get()` member function.

The equality and relational operators for `shared_ptr` only compare the stored pointer returned by `get()`, not the owned pointer. To test whether two `shared_ptr` objects share ownership of the same pointer see `std::shared_ptr::owner_before` and `std::owner_less`.

### 5.892.2 Member Typedef Documentation

**element\_type**

```
template<typename _Tp>
using std::shared_ptr< _Tp >::element_type
The type pointed to by the stored pointer, remove_extents_t<_Tp>
```

### 5.892.3 Constructor & Destructor Documentation

**shared\_ptr()** [1/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr () [inline], [constexpr], [noexcept]
Construct an empty shared_ptr.
```

**Postcondition**

```
use_count()==0 && get()==0
```

**shared\_ptr()** [2/14]

```
template<typename _Tp>
std::shared_ptr<_Tp>::shared_ptr (
    const shared_ptr<_Tp> & ) [default], [noexcept]
```

Copy constructor.

**shared\_ptr()** [3/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<_Yp*>>
std::shared_ptr<_Tp>::shared_ptr (
    _Yp * __p) [inline], [explicit]
```

Construct a shared\_ptr that owns the pointer \_\_p.

**Parameters**

|                     |                                                 |
|---------------------|-------------------------------------------------|
| $\leftarrow$<br>__p | A pointer that is convertible to element_type*. |
|---------------------|-------------------------------------------------|

**Postcondition**

```
use_count() == 1 && get() == __p
```

**Exceptions**

|                            |                                  |
|----------------------------|----------------------------------|
| <i>std::bad_alloc</i> , in | which case delete __p is called. |
|----------------------------|----------------------------------|

**shared\_ptr()** [4/14]

```
template<typename _Tp>
template<typename _Yp, typename _Deleter, typename = _Constructible<_Yp*, _Deleter>>
std::shared_ptr<_Tp>::shared_ptr (
    _Yp * __p,
    _Deleter __d) [inline]
```

Construct a shared\_ptr that owns the pointer \_\_p and the deleter \_\_d.

**Parameters**

|                     |            |
|---------------------|------------|
| $\leftarrow$<br>__p | A pointer. |
| $\leftarrow$<br>__d | A deleter. |

**Postcondition**

```
use_count() == 1 && get() == __p
```

**Exceptions**

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <code>std::bad_alloc</code> , in | which case <code>__d(__p)</code> is called. |
|----------------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw  
`__shared_ptr` will release `__p` by calling `__d(__p)`

**shared\_ptr() [5/14]**

```
template<typename _Tp>
template<typename _Deleter>
std::shared_ptr< _Tp >::shared_ptr (
    nullptr_t __p,
    _Deleter __d) [inline]
```

Construct a `shared_ptr` that owns a null pointer and the deleter `__d`.

**Parameters**

|                                       |                          |
|---------------------------------------|--------------------------|
| $\leftrightarrow$<br><code>__p</code> | A null pointer constant. |
| $\leftrightarrow$<br><code>__d</code> | A deleter.               |

**Postcondition**

`use_count() == 1 && get() == __p`

**Exceptions**

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <code>std::bad_alloc</code> , in | which case <code>__d(__p)</code> is called. |
|----------------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw  
The last owner will call `__d(__p)`

**shared\_ptr() [6/14]**

```
template<typename _Tp>
template<typename _Yp, typename _Deleter, typename _Alloc, typename = _Constructible<_Yp*,  $\leftrightarrow$ 
_Deleter, _Alloc>>
std::shared_ptr< _Tp >::shared_ptr (
    _Yp * __p,
    _Deleter __d,
    _Alloc __a) [inline]
```

Construct a `shared_ptr` that owns the pointer `__p` and the deleter `__d`.

**Parameters**

|                                       |               |
|---------------------------------------|---------------|
| $\leftrightarrow$<br><code>__p</code> | A pointer.    |
| $\leftrightarrow$<br><code>__d</code> | A deleter.    |
| $\leftrightarrow$<br><code>__a</code> | An allocator. |

**Postcondition**

use\_count() == 1 && get() == \_\_p

**Exceptions**

|                            |                                |
|----------------------------|--------------------------------|
| <i>std::bad_alloc</i> , in | which case __d(__p) is called. |
|----------------------------|--------------------------------|

Requirements: \_Deleter's copy constructor and destructor must not throw \_Alloc's copy constructor and destructor must not throw.

\_\_shared\_ptr will release \_\_p by calling \_\_d(\_\_p)

**shared\_ptr()** [7/14]

```
template<typename _Tp>
template<typename _Deleter, typename _Alloc>
std::shared_ptr< _Tp >::shared_ptr (
    nullptr_t __p,
    _Deleter __d,
    _Alloc __a) [inline]
```

Construct a shared\_ptr that owns a null pointer and the deleter \_\_d.

**Parameters**

|                          |                          |
|--------------------------|--------------------------|
| $\leftrightarrow$<br>__p | A null pointer constant. |
| $\leftrightarrow$<br>__d | A deleter.               |
| $\leftrightarrow$<br>__a | An allocator.            |

**Postcondition**

use\_count() == 1 && get() == \_\_p

**Exceptions**

|                            |                                |
|----------------------------|--------------------------------|
| <i>std::bad_alloc</i> , in | which case __d(__p) is called. |
|----------------------------|--------------------------------|

Requirements: \_Deleter's copy constructor and destructor must not throw \_Alloc's copy constructor and destructor must not throw.

The last owner will call \_\_d(\_\_p)

**shared\_ptr()** [8/14]

```
template<typename _Tp>
template<typename _Yp>
std::shared_ptr< _Tp >::shared_ptr (
    const shared_ptr< _Yp > & __r,
    element_type * __p) [inline], [noexcept]
```

Constructs a shared\_ptr instance that stores \_\_p and shares ownership with \_\_r.

**Parameters**

|                                |                                                                    |
|--------------------------------|--------------------------------------------------------------------|
| $\swarrow$<br><code>__r</code> | A <code>shared_ptr</code> .                                        |
| $\swarrow$<br><code>__p</code> | A pointer that will remain valid while <code>*__r</code> is valid. |

**Postcondition**

```
get() == __p && use_count() == __r.use_count()
```

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`. The complete object will remain valid while any `shared_ptr` owns it, even if they don't store a pointer to the complete object.

```
shared_ptr<pair<int,int>> pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

**shared\_ptr() [9/14]**

```
template<typename _Tp>
template<typename _Yp>
std::shared_ptr<_Tp>::shared_ptr (
    shared_ptr<_Yp> && __r,
    element_type * __p) [inline], [noexcept]
```

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

**Parameters**

|                                |                                                                    |
|--------------------------------|--------------------------------------------------------------------|
| $\swarrow$<br><code>__r</code> | A <code>shared_ptr</code> .                                        |
| $\swarrow$<br><code>__p</code> | A pointer that will remain valid while <code>*__r</code> is valid. |

**Postcondition**

```
get() == __p && !__r.use_count() && !__r.get()
```

**Since**

C++17

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`. The complete object will remain valid while any `shared_ptr` owns it, even if they don't store a pointer to the complete object.

```
shared_ptr<pair<int,int>> pii(new pair<int,int>());
shared_ptr<int> pi1(pii, &pii->first);
assert(pii.use_count() == 2);
shared_ptr<int> pi2(std::move(pii), &pii->second);
assert(pii.use_count() == 0);
```

**shared\_ptr() [10/14]**

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>
std::shared_ptr<_Tp>::shared_ptr (
    const shared_ptr<_Yp> & __r) [inline], [noexcept]
```

If `__r` is empty, constructs an empty `shared_ptr`; otherwise construct a `shared_ptr` that shares ownership with `__r`.

## Parameters

|          |               |
|----------|---------------|
| ↩        | A shared_ptr. |
| ↩        |               |
| ↩        |               |
| ↩        |               |
| <i>r</i> |               |

## Postcondition

get() == \_\_r.get() && use\_count() == \_\_r.use\_count()

**shared\_ptr()** [11/14]

```
template<typename _Tp>
std::shared_ptr<_Tp>::shared_ptr (
    shared_ptr<_Tp> && __r) [inline], [noexcept]
```

Move-constructs a shared\_ptr instance from \_\_r.

## Parameters

|          |                      |
|----------|----------------------|
| ↩        | A shared_ptr rvalue. |
| ↩        |                      |
| ↩        |                      |
| ↩        |                      |
| <i>r</i> |                      |

## Postcondition

\*this contains the old value of \_\_r, \_\_r is empty.

**shared\_ptr()** [12/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<shared_ptr<_Yp>>>
std::shared_ptr<_Tp>::shared_ptr (
    shared_ptr<_Yp> && __r) [inline], [noexcept]
```

Move-constructs a shared\_ptr instance from \_\_r.

## Parameters

|          |                      |
|----------|----------------------|
| ↩        | A shared_ptr rvalue. |
| ↩        |                      |
| ↩        |                      |
| ↩        |                      |
| <i>r</i> |                      |

## Postcondition

\*this contains the old value of \_\_r, \_\_r is empty.



**shared\_ptr()** [13/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>
std::shared_ptr< _Tp >::shared_ptr (
    const weak_ptr< _Yp > & __r) [inline], [explicit]
```

Constructs a shared\_ptr that shares ownership with \_\_r and stores a copy of the pointer stored in \_\_r.

**Parameters**

|     |             |
|-----|-------------|
| ↔   | A weak_ptr. |
| __↔ |             |
| ↔   |             |
| __↔ |             |
| r   |             |

**Postcondition**

```
use_count() == __r.use_count()
```

**Exceptions**

|                     |                                                                  |
|---------------------|------------------------------------------------------------------|
| <i>bad_weak_ptr</i> | when __r.expired(), in which case the constructor has no effect. |
|---------------------|------------------------------------------------------------------|

**shared\_ptr()** [14/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
    nullptr_t ) [inline], [constexpr], [noexcept]
```

Construct an empty shared\_ptr.

**Postcondition**

```
use_count() == 0 && get() == nullptr
```

**5.892.4 Member Function Documentation****get()**

```
template<typename _Tp, _Lock_policy _Lp>
element_type * std::__shared_ptr< _Tp, _Lp >::get () const [inline], [noexcept], [inherited]
```

Return the stored pointer.

**operator bool()**

```
template<typename _Tp, _Lock_policy _Lp>
std::__shared_ptr< _Tp, _Lp >::operator bool () const [inline], [explicit], [noexcept], [inherited]
```

Return true if the stored pointer is not null.

**owner\_before()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1>
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
    __shared_ptr< _Tp1, _Lp > const & __rhs) const [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.

This function defines a strict weak ordering between two `shared_ptr` or `weak_ptr` objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

#### owner\_before() [2/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1>
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
    __weak_ptr< _Tp1, _Lp > const & __rhs) const [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.

This function defines a strict weak ordering between two `shared_ptr` or `weak_ptr` objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

#### swap()

```
template<typename _Tp, _Lock_policy _Lp>
void std::__shared_ptr< _Tp, _Lp >::swap (
    __shared_ptr< _Tp, _Lp > & __other) [inline], [noexcept], [inherited]
```

Exchange both the owned pointer and the stored pointer.

#### unique()

```
template<typename _Tp, _Lock_policy _Lp>
bool std::__shared_ptr< _Tp, _Lp >::unique () const [inline], [noexcept], [inherited]
```

Return true if `use_count() == 1`.

#### use\_count()

```
template<typename _Tp, _Lock_policy _Lp>
long std::__shared_ptr< _Tp, _Lp >::use_count () const [inline], [noexcept], [inherited]
```

If `*this` owns a pointer, return the number of owners, otherwise zero.

The documentation for this class was generated from the following files:

- [bits/shared\\_ptr.h](#)
- [auto\\_ptr.h](#)

## 5.893 std::shared\_timed\_mutex Class Reference

```
#include <shared_mutex>
```

### Public Member Functions

- `shared_timed_mutex` (const `shared_timed_mutex` &)=delete
- void `lock` ()
- void `lock_shared` ()
- `shared_timed_mutex` & `operator=` (const `shared_timed_mutex` &)=delete
- bool `try_lock` ()
- template<typename `_Rep`, typename `_Period`>  
bool `try_lock_for` (const `chrono::duration`< `_Rep`, `_Period` > &\_\_rtime)
- bool `try_lock_shared` ()
- template<typename `_Rep`, typename `_Period`>  
bool `try_lock_shared_for` (const `chrono::duration`< `_Rep`, `_Period` > &\_\_rtime)
- template<typename `_Clock`, typename `_Duration`>  
bool `try_lock_shared_until` (const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_abs\_time)

- `template<typename _Clock, typename _Duration>`  
`bool try_lock_until (const chrono::time_point< _Clock, _Duration > &__abs_time)`
- `void unlock ()`
- `void unlock_shared ()`

### 5.893.1 Detailed Description

The standard shared timed mutex type.

The documentation for this class was generated from the following file:

- [shared\\_mutex](#)

## 5.894 std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k > Class Template Reference

```
#include <random>
```

### Public Types

- `template<typename _Sseq>`  
`using _If_seed_seq`
- `typedef _RandomNumberEngine::result_type result_type`

### Public Member Functions

- [shuffle\\_order\\_engine](#) ()
- [shuffle\\_order\\_engine](#) (\_RandomNumberEngine &&\_\_rng)
- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`  
[shuffle\\_order\\_engine](#) (\_Sseq &\_\_q)
- [shuffle\\_order\\_engine](#) (const \_RandomNumberEngine &\_\_rng)
- [shuffle\\_order\\_engine](#) (result\_type \_\_s)
- `const _RandomNumberEngine & base () const` noexcept
- `void discard` (unsigned long long \_\_z)
- [result\\_type operator\(\)](#) ()
- `void seed` ()
- `template<typename _Sseq>`  
`_If_seed_seq<_Sseq > seed` (\_Sseq &\_\_q)
- `void seed` (result\_type \_\_s)

### Static Public Member Functions

- `static constexpr result_type max` ()
- `static constexpr result_type min` ()

### Static Public Attributes

- `static constexpr size_t table_size`

## Friends

- template<typename \_RandomNumberEngine1, size\_t \_\_k1, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const std::shuffle\_order\_engine< \_RandomNumberEngine1, \_\_k1 > &\_\_x)
- bool operator== (const shuffle\_order\_engine &\_\_lhs, const shuffle\_order\_engine &\_\_rhs)
- template<typename \_RandomNumberEngine1, size\_t \_\_k1, typename \_CharT, typename \_Traits>  
std::basic\_istream< \_CharT, \_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is, std::shuffle\_order\_engine< \_RandomNumberEngine1, \_\_k1 > &\_\_x)

### 5.894.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __k>
class std::shuffle_order_engine< _RandomNumberEngine, __k >
```

Produces random numbers by reordering random numbers from some base engine.

The values from the base engine are stored in a sequence of size \_\_k and shuffled by an algorithm that depends on those values.

Since

C++11

### 5.894.2 Member Typedef Documentation

#### result\_type

```
template<typename _RandomNumberEngine, size_t __k>
typedef _RandomNumberEngine::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >↔
::result_type
```

The type of the generated random value.

### 5.894.3 Constructor & Destructor Documentation

#### shuffle\_order\_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine () [inline]
```

Constructs a default shuffle\_order\_engine engine.

The underlying engine is default constructed as well.

#### shuffle\_order\_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
    const _RandomNumberEngine & __rng) [inline], [explicit]
```

Copy constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

**shuffle\_order\_engine()** [3/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
    _RandomNumberEngine && __rng) [inline], [explicit]
```

Move constructs a `shuffle_order_engine` engine.

Copies an existing base class random number generator.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

**shuffle\_order\_engine()** [4/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
    result_type __s) [inline], [explicit]
```

Seed constructs a `shuffle_order_engine` engine.

Constructs the underlying generator engine seeded with `__s`.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A seed value for the base class engine. |
|------------------|-----------------------------------------|

**shuffle\_order\_engine()** [5/5]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
    _Sseq & __q) [inline], [explicit]
```

Generator construct a `shuffle_order_engine` engine.

**Parameters**

|                  |                  |
|------------------|------------------|
| <code>__q</code> | A seed sequence. |
|------------------|------------------|

**5.894.4 Member Function Documentation****base()**

```
template<typename _RandomNumberEngine, size_t __k>
const _RandomNumberEngine & std::shuffle_order_engine< _RandomNumberEngine, __k >::base () const
[inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

Referenced by `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::seed()`.

**discard()**

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::discard (
    unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

**max()**

```
template<typename _RandomNumberEngine, size_t __k>
static constexpr result\_type std::shuffle\_order\_engine<_RandomNumberEngine, __k>::max () [inline],
[static], [constexpr]
```

Gets the maximum value in the generated random number range.

**min()**

```
template<typename _RandomNumberEngine, size_t __k>
static constexpr result\_type std::shuffle\_order\_engine<_RandomNumberEngine, __k>::min () [inline],
[static], [constexpr]
```

Gets the minimum value in the generated random number range.

**operator>()()**

```
template<typename _RandomNumberEngine, size_t __k>
shuffle\_order\_engine<_RandomNumberEngine, __k>::result_type std::shuffle\_order\_engine<_↵
RandomNumberEngine, __k>::operator() ()
```

Gets the next value in the generated random number sequence.

References [std::max\(\)](#), and [std::min\(\)](#).

**seed()** [1/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle\_order\_engine<_RandomNumberEngine, __k>::seed () [inline]
```

Reseeds the `shuffle_order_engine` object with the default seed for the underlying base class generator engine.

**seed()** [2/3]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq>
_If_seed_seq<_Sseq> std::shuffle\_order\_engine<_RandomNumberEngine, __k>::seed (
    _Sseq & __q) [inline]
```

Reseeds the `shuffle_order_engine` object with the given seed sequence.

**Parameters**

|                    |                            |
|--------------------|----------------------------|
| <a href="#">_↵</a> | A seed generator function. |
| <a href="#">_q</a> |                            |

**seed()** [3/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle\_order\_engine<_RandomNumberEngine, __k>::seed (
    result\_type __s) [inline]
```

Reseeds the `shuffle_order_engine` object with the default seed for the underlying base class generator engine.

**5.894.5 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
std::basic\_ostream<_CharT, _Traits> & operator<< (
```

```
std::basic_ostream< _CharT, _Traits > & __os,
const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x) [friend]
```

Inserts the current state of a shuffle\_order\_engine random number generator engine \_\_x into the output stream \_\_os.

#### Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A shuffle_order_engine random number generator engine. |

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

#### operator==

```
template<typename _RandomNumberEngine, size_t __k>
bool operator== (
    const std::shuffle_order_engine< _RandomNumberEngine, __k > & __lhs,
    const std::shuffle_order_engine< _RandomNumberEngine, __k > & __rhs) [friend]
```

Compares two shuffle\_order\_engine random number generator objects of the same type for equality.

#### Parameters

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>__lhs</code> | A shuffle_order_engine random number generator object.       |
| <code>__rhs</code> | Another shuffle_order_engine random number generator object. |

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

#### operator>>

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

#### Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__is</code> | An input stream.                                       |
| <code>__x</code>  | A shuffle_order_engine random number generator engine. |

#### Returns

The input stream with the state of \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.895 std::slice Class Reference

```
#include <slice_array.h>
```

### Public Member Functions

- [slice](#) ()
- [slice](#) (size\_t \_\_o, size\_t \_\_d, size\_t \_\_s)
- size\_t [size](#) () const
- size\_t [start](#) () const
- size\_t [stride](#) () const

### Friends

- bool [operator==](#) (const [slice](#) &, const [slice](#) &)=default

### 5.895.1 Detailed Description

Class defining one-dimensional subset of an array.

The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

### 5.895.2 Friends And Related Symbol Documentation

#### [operator==](#)

```
bool operator== (
    const slice & ,
    const slice & ) [friend]
```

Equality comparison.

References [slice\(\)](#).

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

## 5.896 std::slice\_array< \_Tp > Class Template Reference

```
#include <slice_array.h>
```

### Public Types

- typedef \_Tp **value\_type**

### Public Member Functions

- [slice\\_array](#) (const [slice\\_array](#) &)
- template<class \_Dom>  
void **operator%=>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator%=>** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom>  
void **operator&=>** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator&=>** (const [valarray](#)< \_Tp > &) const



- `template<class _Dom>`  
`void operator*=(const _Expr< _Dom, _Tp > &) const`
- `void operator*=(const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator+=(const _Expr< _Dom, _Tp > &) const`
- `void operator+=(const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator-=(const _Expr< _Dom, _Tp > &) const`
- `void operator-=(const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator/=(const _Expr< _Dom, _Tp > &) const`
- `void operator/=(const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator<<=(const _Expr< _Dom, _Tp > &) const`
- `void operator<<=(const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator=(const _Expr< _Dom, _Tp > &) const`
- `void operator=(const _Tp &) const`
- `slice_array & operator=(const slice_array &)`
- `void operator=(const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator>>=(const _Expr< _Dom, _Tp > &) const`
- `void operator>>=(const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator^=(const _Expr< _Dom, _Tp > &) const`
- `void operator^=(const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator|=(const _Expr< _Dom, _Tp > &) const`
- `void operator|=(const valarray< _Tp > &) const`

## Friends

- `class valarray< _Tp >`

### 5.896.1 Detailed Description

`template<typename _Tp>`  
`class std::slice_array< _Tp >`

Reference to one-dimensional subset of an array.

A `slice_array` is a reference to the actual elements of an array specified by a slice. The way to get a `slice_array` is to call `operator[]`(slice) on a `valarray`. The returned `slice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `slice_array` refers to.

#### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

## 5.896.2 Member Function Documentation

### operator%=( )

```
template<typename _Tp>
void std::slice_array<_Tp>::operator%= (
    const valarray<_Tp> & ) const
```

Modulo slice elements by corresponding elements of *v*.

### operator&=( )

```
template<typename _Tp>
void std::slice_array<_Tp>::operator&= (
    const valarray<_Tp> & ) const
```

Logical and slice elements with corresponding elements of *v*.

### operator\*=( )

```
template<typename _Tp>
void std::slice_array<_Tp>::operator*= (
    const valarray<_Tp> & ) const
```

Multiply slice elements by corresponding elements of *v*.

### operator+=( )

```
template<typename _Tp>
void std::slice_array<_Tp>::operator+= (
    const valarray<_Tp> & ) const
```

Add corresponding elements of *v* to slice elements.

### operator-=( )

```
template<typename _Tp>
void std::slice_array<_Tp>::operator-= (
    const valarray<_Tp> & ) const
```

Subtract corresponding elements of *v* from slice elements.

### operator/=( )

```
template<typename _Tp>
void std::slice_array<_Tp>::operator/= (
    const valarray<_Tp> & ) const
```

Divide slice elements by corresponding elements of *v*.

### operator<<=( )

```
template<typename _Tp>
void std::slice_array<_Tp>::operator<<= (
    const valarray<_Tp> & ) const
```

Left shift slice elements by corresponding elements of *v*.

### operator>>=( )

```
template<typename _Tp>
void std::slice_array<_Tp>::operator>>= (
    const valarray<_Tp> & ) const
```

Right shift slice elements by corresponding elements of *v*.  
References [slice\\_array\(\)](#).

### **operator^=()**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator^= (
    const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

### **operator" |=()**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator|= (
    const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following files:

- [valarray](#)
- [slice\\_array.h](#)

## **5.897 \_\_gnu\_cxx::slist< \_Tp, \_Alloc > Class Template Reference**

```
#include <slist>
```

### **Public Types**

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_Slist\_iterator< \_Tp, const \_Tp &, const \_Tp \* > **const\_iterator**
- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Slist\_iterator< \_Tp, \_Tp &, \_Tp \* > **iterator**
- typedef value\_type \* **pointer**
- typedef value\_type & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

### **Public Member Functions**

- template<class \_InputIterator>  
  **slist** ( \_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type & \_\_a=allocator\_type())
- **slist** (const allocator\_type & \_\_a=allocator\_type())
- **slist** (const [slist](#) & \_\_x)
- **slist** (size\_type \_\_n)
- **slist** (size\_type \_\_n, const value\_type & \_\_x, const allocator\_type & \_\_a=allocator\_type())
- template<class \_InputIterator>  
  void **\_M\_assign\_dispatch** ( \_InputIterator \_\_first, \_InputIterator \_\_last, std::\_\_false\_type)
- template<class \_Integer>  
  void **\_M\_assign\_dispatch** ( \_Integer \_\_n, \_Integer \_\_val, std::\_\_true\_type)
- void **\_M\_fill\_assign** (size\_type \_\_n, const \_Tp & \_\_val)
- template<class \_InputIterator>  
  void **assign** ( \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const \_Tp & \_\_val)

- iterator **before\_begin** ()
- const\_iterator **before\_begin** () const
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_pos)
- iterator **erase\_after** (iterator \_\_before\_first, iterator \_\_last)
- iterator **erase\_after** (iterator \_\_pos)
- reference **front** ()
- const\_reference **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (iterator \_\_pos)
- template<class \_InIterator>  
void **insert** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- iterator **insert** (iterator \_\_pos, const value\_type &\_\_x)
- void **insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- iterator **insert\_after** (iterator \_\_pos)
- template<class \_InIterator>  
void **insert\_after** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- iterator **insert\_after** (iterator \_\_pos, const value\_type &\_\_x)
- void **insert\_after** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- size\_type **max\_size** () const
- template<class \_StrictWeakOrdering>  
void **merge** (slist &, \_StrictWeakOrdering)
- void **merge** (slist &\_\_x)
- slist & **operator=** (const slist &\_\_x)
- void **pop\_front** ()
- iterator **previous** (const\_iterator \_\_pos)
- const\_iterator **previous** (const\_iterator \_\_pos) const
- void **push\_front** ()
- void **push\_front** (const value\_type &\_\_x)
- void **remove** (const \_Tp &\_\_val)
- template<class \_Predicate>  
void **remove\_if** (\_Predicate \_\_pred)
- void **resize** (size\_type new\_size)
- void **resize** (size\_type new\_size, const \_Tp &\_\_x)
- void **reverse** ()
- size\_type **size** () const
- void **sort** ()
- template<class \_StrictWeakOrdering>  
void **sort** (\_StrictWeakOrdering \_\_comp)
- void **splice** (iterator \_\_pos, slist &\_\_x)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_i)
- void **splice\_after** (iterator \_\_pos, iterator \_\_before\_first, iterator \_\_before\_last)
- void **splice\_after** (iterator \_\_pos, iterator \_\_prev)
- void **splice\_after** (iterator \_\_pos, slist &\_\_x)

- void **swap** ([slist](#) &\_\_x)
- void **unique** ()
- template<class \_BinaryPredicate>  
void **unique** (\_BinaryPredicate \_\_pred)

### Static Private Member Functions

- static constexpr size\_type [max\\_size](#) (const \_Alloc &\_\_a) noexcept

#### 5.897.1 Detailed Description

```
template<class _Tp, class _Alloc = std::allocator<_Tp>>  
class __gnu_cxx::slist<_Tp, _Alloc >
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

The documentation for this class was generated from the following file:

- [slist](#)

### 5.898 std::experimental::filesystem::v1::space\_info Struct Reference

```
#include <fs_fwd.h>
```

#### Public Attributes

- uintmax\_t **available**
- uintmax\_t **capacity**
- uintmax\_t **free**

#### 5.898.1 Detailed Description

Information about free space on a disk.

The documentation for this struct was generated from the following file:

- [experimental/bits/fs\\_fwd.h](#)

### 5.899 std::filesystem::space\_info Struct Reference

```
#include <fs_fwd.h>
```

#### Public Attributes

- uintmax\_t **available**
- uintmax\_t **capacity**
- uintmax\_t **free**

#### Friends

- bool **operator==** (const [space\\_info](#) &, const [space\\_info](#) &)=default

### 5.899.1 Detailed Description

Information about free space on a disk.

The documentation for this struct was generated from the following file:

- [bits/fs\\_fwd.h](#)

## 5.900 `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

```
#include <splay_tree_.hpp>
```

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `base_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `splay_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key\_const\_pointer**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_pointer` **key\_pointer**
- typedef `base_type::key_reference` **key\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `base_type::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `base_type::mapped_const_reference` **mapped\_const\_reference**
- typedef `base_type::mapped_pointer` **mapped\_pointer**
- typedef `base_type::mapped_reference` **mapped\_reference**
- typedef `base_type::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**
- typedef `base_type::node_update` **node\_update**
- typedef `base_type::const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `base_type::reverse_iterator` **reverse\_iterator**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `base_type::value_type` **value\_type**

## Public Member Functions

- **splay\_tree\_map** (const Cmp\_Fn &)
- **splay\_tree\_map** (const Cmp\_Fn &, const node\_update &)
- **splay\_tree\_map** (const [splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It>  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator it)
- bool **erase** (key\_const\_reference)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **initialize** ()
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference r\_value)
- void **join** ([splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_iterator [node\\_begin](#) ()
- node\_const\_iterator [node\\_begin](#) () const
- node\_iterator [node\\_end](#) ()
- node\_const\_iterator [node\\_end](#) () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, [splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** ([splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Protected Types

- typedef `node_alloc_traits::value_type` **node**
- typedef `node_alloc_traits::allocator_type` **node\_allocator**
- typedef `traits_type::null_node_update_pointer` **null\_node\_update\_pointer**
- typedef `types_traits< Key, Mapped, _Alloc, false >` **traits\_base**

## Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- template<typename Node\_Update\_>  
void **apply\_update** (node\_pointer, Node\_Update\_\*)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- `std::pair< node_pointer, bool >` **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- `std::pair< point_iterator, bool >` **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_subtree\_size** (node\_pointer)
- template<typename Node\_Update\_>  
void **update\_to\_top** (node\_pointer, Node\_Update\_\*)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

## Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

## Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

## Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

### 5.900.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename \_Alloc>

class `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`

Splay tree.



## 5.900.2 Member Function Documentation

### node\_begin() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin
() [inline], [inherited]
```

Returns a `node_iterator` corresponding to the node at the root of the tree.

### node\_begin() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↵
begin () const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to the node at the root of the tree.

### node\_end() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()
[inline], [inherited]
```

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

### node\_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↵
end () const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- [splay\\_tree.hpp](#)

## 5.901 \_\_gnu\_pbds::detail::splay\_tree\_node\_< Value\_Type, Metadata, \_Alloc > Struct Template Reference

```
#include <node.hpp>
```

### Public Types

- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::const\_reference **metadata\_const\_reference**
- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::reference **metadata\_reference**
- typedef Metadata **metadata\_type**
- typedef [rebind\\_traits](#)< \_Alloc, [splay\\_tree\\_node\\_](#) >::pointer **node\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, [splay\\_tree\\_node\\_](#) >::size\_type **size\_type**
- typedef Value\_Type **value\_type**

### Public Member Functions

- metadata\_reference **get\_metadata** ()
- metadata\_const\_reference **get\_metadata** () const
- bool **special** () const

### Public Attributes

- metadata\_type **m\_metadata**
- node\_pointer **m\_p\_left**
- node\_pointer **m\_p\_parent**
- node\_pointer **m\_p\_right**
- bool **m\_special**
- size\_type **m\_subtree\_size**
- value\_type **m\_value**

#### 5.901.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>  
struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >
```

Node for splay tree.

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/node.hpp](#)

### 5.902 \_\_gnu\_pbds::splay\_tree\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::splay_tree_tag`:



### 5.902.1 Detailed Description

Splay tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.903 `std::stack<_Tp, _Sequence>` Class Template Reference

```
#include <std_stack.h>
```

### Public Types

- `typedef _Sequence::const_reference` **const\_reference**
- `typedef _Sequence` **container\_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size\_type**
- `typedef _Sequence::value_type` **value\_type**

### Public Member Functions

- `template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type>`  
[stack](#) ()

- **stack** (`_Sequence &&__c`)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (`_Sequence &&__c, const _Alloc &__a`)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (`const _Alloc &__a`)
- **stack** (`const _Sequence &__c`)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (`const _Sequence &__c, const _Alloc &__a`)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (`const stack &__q, const _Alloc &__a`)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (`stack &&__q, const _Alloc &__a`)
- `template<typename... _Args>`  
`decltype(auto) emplace (_Args &&... __args)`
- `bool empty () const`
- `void pop ()`
- `void push (const value_type &__x)`
- `void push (value_type &&__x)`
- `size_type size () const`
- `void swap (stack &__s) noexcept(__is_nothrow_swappable<_Sequence>::value)`
- `reference top ()`
- `const_reference top () const`

### Protected Attributes

- `_Sequence c`

### Friends

- `template<typename _Tp1, typename _Seq1>`  
`bool operator< (const stack<_Tp1, _Seq1> &, const stack<_Tp1, _Seq1> &)`
- `template<typename _Tp1, three_way_comparable _Seq1>`  
`compare\_three\_way\_result\_t<_Seq1> operator<=> (const stack<_Tp1, _Seq1> &, const stack<_Tp1, _Seq1> &)`
- `template<typename _Tp1, typename _Seq1>`  
`bool operator== (const stack<_Tp1, _Seq1> &, const stack<_Tp1, _Seq1> &)`

#### 5.903.1 Detailed Description

`template<typename _Tp, typename _Sequence = deque<_Tp>>`  
**class** `std::stack<_Tp, _Sequence>`

A standard container giving FILO behavior.

#### Template Parameters

|                        |                                                                          |
|------------------------|--------------------------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                                         |
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> . |

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `back`, `push_back`, and `pop_back`, such as `std::list`, `std::vector`, or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

### 5.903.2 Constructor & Destructor Documentation

#### **stack()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type>
```

```
std::stack< _Tp, _Sequence >::stack () [inline]
```

Default constructor creates no elements.

### 5.903.3 Member Function Documentation

#### **empty()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::stack< _Tp, _Sequence >::empty () const [inline], [nodiscard]
```

Returns true if the stack is empty.

#### **pop()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::pop () [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

#### **push()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::push (
    const value_type & __x) [inline]
```

Add data to the top of the stack.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

#### **size()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::stack< _Tp, _Sequence >::size () const [inline], [nodiscard]
```

Returns the number of elements in the stack.

**top()** [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::stack<_Tp, _Sequence>::top () [inline], [nodiscard]
Returns a read/write reference to the data at the first element of the stack.
```

**top()** [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::stack<_Tp, _Sequence>::top () const [inline], [nodiscard]
Returns a read-only (constant) reference to the data at the first element of the stack.
The documentation for this class was generated from the following file:
```

- [stl\\_stack.h](#)

**5.904 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference**

```
#include <stdio_filebuf.h>
```

Inheritance diagram for `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`:

**Public Types**

- `typedef codecvt< char_type, char, __state_type > __codecvt_type`
- `typedef __basic_file< char > __file_type`
- `typedef basic\_filebuf< char_type, traits_type > __filebuf_type`
- `typedef traits_type::state_type __state_type`
- `typedef basic\_streambuf< char_type, traits_type > __streambuf_type`
- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`

- typedef std::size\_t **size\_t**
- typedef \_Traits **traits\_type**

### Public Member Functions

- [stdio\\_filebuf](#) ()
- [stdio\\_filebuf](#) (int \_\_fd, [std::ios\\_base::openmode](#) \_\_mode, size\_t \_\_size=static\_cast< size\_t >(BUFSIZ))
- [stdio\\_filebuf](#) (std::\_\_c\_file \* \_\_f, [std::ios\\_base::openmode](#) \_\_mode, size\_t \_\_size=static\_cast< size\_t >(BUFSIZ))
- [stdio\\_filebuf](#) ([stdio\\_filebuf](#) &&)=default
- virtual ~[stdio\\_filebuf](#) ()
- [\\_\\_filebuf\\_type](#) \* [close](#) ()
- int [fd](#) ()
- std::\_\_c\_file \* [file](#) ()
- locale [getloc](#) () const
- streamsize [in\\_avail](#) ()
- bool [is\\_open](#) () const throw ()
- template<typename \_Path>  
  [\\_If\\_fs\\_path](#)< \_Path, [\\_\\_filebuf\\_type](#) \* > [open](#) (const \_Path & \_\_s, ios\_base::openmode \_\_mode)
- [\\_\\_filebuf\\_type](#) \* [open](#) (const char \* \_\_s, ios\_base::openmode \_\_mode)
- [\\_\\_filebuf\\_type](#) \* [open](#) (const std::string & \_\_s, ios\_base::openmode \_\_mode)
- [stdio\\_filebuf](#) & **operator=** ([stdio\\_filebuf](#) &&)=default
- locale [pubimbue](#) (const locale & \_\_loc)
- int\_type [sbumpc](#) ()
- int\_type [sgetc](#) ()
- streamsize [sgetn](#) (char\_type \* \_\_s, streamsize \_\_n)
- int\_type [snextc](#) ()
- int\_type [sputbackc](#) (char\_type \_\_c)
- int\_type [sputc](#) (char\_type \_\_c)
- streamsize [sputn](#) (const char\_type \* \_\_s, streamsize \_\_n)
- int\_type [sungetc](#) ()
- void **swap** ([stdio\\_filebuf](#) & \_\_fb)
- void **swap** ([basic\\_filebuf](#) &)
  
- [basic\\_streambuf](#) \* [pubsetbuf](#) (char\_type \* \_\_s, streamsize \_\_n)
- pos\_type [pubseekoff](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- pos\_type [pubseekpos](#) (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- int [pubsync](#) ()

### Protected Member Functions

- void [\\_\\_safe\\_gbump](#) (streamsize \_\_n)
- void [\\_\\_safe\\_pbump](#) (streamsize \_\_n)
- void [\\_M\\_allocate\\_internal\\_buffer](#) ()
- bool [\\_M\\_convert\\_to\\_external](#) (char\_type \*, streamsize)
- void [\\_M\\_create\\_pback](#) ()
- void [\\_M\\_destroy\\_internal\\_buffer](#) () throw ()
- void [\\_M\\_destroy\\_pback](#) () throw ()
- int [\\_M\\_get\\_ext\\_pos](#) (\_\_state\_type & \_\_state)
- pos\_type [\\_M\\_seek](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, \_\_state\_type \_\_state)
- void [\\_M\\_set\\_buffer](#) (streamsize \_\_off)

- `bool _M_terminate_output ()`
  - `void gbump (int __n)`
  - `virtual void imbue (const locale & __loc)`
  - `virtual int_type overflow (int_type __c=_Traits::eof())`
  - `virtual int_type pbackfail (int_type __c=_Traits::eof())`
  - `void pbump (int __n)`
  - `virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `virtual __streambuf_type * setbuf (char_type * __s, streamsize __n)`
  - `void setg (char_type * __gbeg, char_type * __gnext, char_type * __gend)`
  - `void setp (char_type * __pbeg, char_type * __pend)`
  - `virtual streamsize showmanyc ()`
  - `void swap (basic_streambuf & __sb)`
  - `virtual int sync ()`
  - `virtual int_type uflow ()`
  - `virtual int_type underflow ()`
  - `virtual streamsize xsgetn (char_type * __s, streamsize __n)`
  - `virtual streamsize xsputn (const char_type * __s, streamsize __n)`
- 
- `char_type * eback () const`
  - `char_type * gptr () const`
  - `char_type * egptr () const`
- 
- `char_type * pbase () const`
  - `char_type * pptr () const`
  - `char_type * epptr () const`

### Protected Attributes

- `char_type * _M_buf`
- `bool _M_buf_allocated`
- `locale _M_buf_locale`
- `size_t _M_buf_size`
- `const __codecvt_type * _M_codecvt`
- `char * _M_ext_buf`
- `streamsize _M_ext_buf_size`
- `char * _M_ext_end`
- `const char * _M_ext_next`
- `__file_type _M_file`
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `__c_lock _M_lock`
- `ios_base::openmode _M_mode`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- `bool _M_reading`



- `__state_type _M_state_beg`
- `__state_type _M_state_cur`
- `__state_type _M_state_last`
- `bool _M_writing`
- `char_type _M_pback`
- `char_type * _M_pback_cur_save`
- `char_type * _M_pback_end_save`
- `bool _M_pback_init`

#### 5.904.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

#### 5.904.2 Constructor & Destructor Documentation

##### `stdio_filebuf()` [1/3]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf () [inline]
deferred initialization
References std::basic\_filebuf< \_CharT, \_Traits >::basic\_filebuf\(\).
```

##### `stdio_filebuf()` [2/3]

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
    int __fd,
    std::ios_base::openmode __mode,
    size_t __size = static_cast<size_t>( BUFSIZ ))
```

##### Parameters

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <code>__fd</code>   | An open file descriptor.                                |
| <code>__mode</code> | Same meaning as in a standard filebuf.                  |
| <code>__size</code> | Optimal or preferred size of internal buffer, in chars. |

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

References [std::basic\\_filebuf< \\_CharT, \\_Traits >::M\\_buf\\_size](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::M\\_mode](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::M\\_reading](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::M\\_set\\_buffer\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::is\\_open\(\)](#).

##### `stdio_filebuf()` [3/3]

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
    std::__c_file * __f,
    std::ios_base::openmode __mode,
    size_t __size = static_cast<size_t>( BUFSIZ ))
```

## Parameters

|                     |                                                                                                    |
|---------------------|----------------------------------------------------------------------------------------------------|
| <code>__f</code>    | An open <code>FILE*</code> .                                                                       |
| <code>__mode</code> | Same meaning as in a standard filebuf.                                                             |
| <code>__size</code> | Optimal or preferred size of internal buffer, in chars. Defaults to system's <code>BUFSIZ</code> . |

This constructor associates a file stream buffer with an open C `FILE*`. The `FILE*` will not be automatically closed when the `stdio_filebuf` is closed/destroyed.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

`~stdio_filebuf()`

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf<_CharT, _Traits>::~~stdio_filebuf () [virtual]
```

Closes the external data stream if the file descriptor constructor was used.

## 5.904.3 Member Function Documentation

`_M_create_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_create_pback () [inline], [protected], [inherited]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back  
Referenced by `pbackfail()`.

`_M_destroy_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback () throw () [inline], [protected], [inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.  
Referenced by `overflow()`, `seekoff()`, `seekpos()`, `underflow()`, and `xsgetn()`.

`_M_set_buffer()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_set_buffer (
    streamsize __off) [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `egptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Referenced by `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, `close()`, `imbue()`, `open()`, `overflow()`, `pbackfail()`, `underflow()`, `xsgetn()`, and `xspn()`.

`close()`

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::~__filebuf_type * std::basic_filebuf<_CharT, _Traits>::close
() [inherited]
```

Closes the currently associated file.

## Returns

this on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

References [basic\\_filebuf\(\)](#), [\\_M\\_mode](#), [\\_M\\_pback\\_init](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), and [is\\_open\(\)](#).

Referenced by [open\(\)](#).

## eback()

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic\_streambuf< \_CharT, \_Traits >::eback \(\) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::pbackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

## egptr()

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic\_streambuf< \_CharT, \_Traits >::egptr \(\) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::underflow\(\)](#), [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >::underflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#), and [xsgetn\(\)](#).

## epptr()

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic\_streambuf< \_CharT, \_Traits >::epptr \(\) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**fd()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
int __gnu_cxx::stdio_filebuf<_CharT, _Traits>::fd () [inline]
```

**Returns**

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

**file()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
std::__c_file * __gnu_cxx::stdio_filebuf<_CharT, _Traits>::file () [inline]
```

**Returns**

The underlying FILE\*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

**gbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::gbump (
    int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Referenced by `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

**gptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

### imbue()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::imbue (
    const locale & __loc) [protected], [virtual], [inherited]
```

Changes translations.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_ext_buf`, `_M_ext_next`, `_M_mode`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `is_open()`, and `seekoff()`.

### in\_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

### is\_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::is_open () const throw () [inline], [nodiscard],
[inherited]
```

Returns true if the external file is open.

Referenced by `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, `close()`, `imbue()`, `open()`, `seekoff()`, `seekpos()`, `setbuf()`, and `showmanyc()`.

**open()** [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, __filebuf_type * > std::basic_filebuf< _CharT, _Traits >::open (
    const _Path & __s,
    ios_base::openmode __mode) [inline], [inherited]
```

Opens an external file.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The name of the file, as a filesystem::path. |
| <code>__mode</code> | The open mode flags.                         |

**Returns**

`this` on success, `NULL` on failure

**open()** [2/3]

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
    const char * __s,
    ios_base::openmode __mode) [inherited]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

**Returns**

`this` on success, `NULL` on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

| ios_base Flag combination |    |     |       |     | stdio equivalent |
|---------------------------|----|-----|-------|-----|------------------|
| binary                    | in | out | trunc | app |                  |
|                           |    | +   |       |     | w                |
|                           |    | +   |       | +   | a                |
|                           |    |     |       | +   | a                |
|                           |    | +   | +     |     | w                |
|                           | +  |     |       |     | r                |
|                           | +  | +   |       |     | r+               |
|                           | +  | +   | +     |     | w+               |
|                           | +  | +   |       | +   | a+               |
|                           |    | +   |       | +   | a+               |
| +                         |    | +   |       |     | wb               |

|         |   |   |   |   |   |  |     |  |
|---------|---|---|---|---|---|--|-----|--|
|         | + |   | + |   | + |  | ab  |  |
|         | + |   |   |   | + |  | ab  |  |
|         | + |   | + |   | + |  | wb  |  |
|         | + | + |   |   |   |  | rb  |  |
|         | + | + | + |   |   |  | r+b |  |
|         | + | + | + | + |   |  | w+b |  |
|         | + | + | + |   | + |  | a+b |  |
|         | + | + |   |   | + |  | a+b |  |
| +-----+ |   |   |   |   |   |  |     |  |

References [\\_M\\_mode](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), [std::ios\\_base::ate](#), [close\(\)](#), [std::ios\\_base::end](#), [is\\_open\(\)](#), and [seekoff\(\)](#).

### open() [3/3]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
    const std::string & __s,
    ios_base::openmode __mode) [inline], [inherited]
```

Opens an external file.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

#### Returns

`this` on success, `NULL` on failure

### overflow()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (
    int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

#### Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns eof().

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_buf_size`, `_M_destroy_pback()`, `_M_mode`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::app`, `std::ios_base::cur`, `std::ios_base::out`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, `std::basic_streambuf<_CharT, _Traits>::pbump()` and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

Referenced by `pbackfail()`, `sync()`, `underflow()`, and `xsgetn()`.

**pbackfail()**

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::pbackfail (
    int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_↵</code> | The character to be inserted back into the sequence. |
| <code>_C</code> |                                                      |

**Returns**

eof() on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns eof().

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_create_pback()`, `_M_mode`, `_M_pback_init`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::gbump()`, `std::basic_streambuf<_CharT, _Traits>::in`, `overflow()`, `seekoff()`, and `underflow()`.

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::sync()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.



**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
    int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#).

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [eptr\(\)](#) returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived [imbue\(\\_\\_loc\)](#).

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

**pubsetbuf()**

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

**pubsync()**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]
```

Calls virtual sync function.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

**sbumpc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`.

**seekoff()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual], [inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_destroy_pback()`, `_M_reading`, `std::ios_base::cur`, `is_open()`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

Referenced by `imbue()`, `open()`, and `pbackfail()`.

**seekpos()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual], [inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_destroy_pback()`, `std::ios_base::beg`, and `is_open()`.

**setbuf()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >↵
::setbuf (
    char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Manipulates the buffer.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <code>↵<br/>__s</code> | Pointer to a buffer area.  |
| <code>↵<br/>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.↵html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_buf`, `_M_buf_size`, and `is_open()`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setp (
    char_type * __pbeg,
    char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`.

**sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::ignore()`.

## sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for xsgetn.

### Parameters

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

## showmanyc()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc () [protected], [virtual], [inherited]
Investigating the data available.
```

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::ios_base::binary`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::ios_base::in`, and `is_open()`.

## snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline], [inherited]
```

Getting the next character.

### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< char_type, traits_type >::ignore()`, `std::basic_istream< char_type, traits_type >::operator>>()`, and `std::basic_istream< char_type, traits_type >::putback()`.

**sputbackc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
    char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::putc (
    char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
    const char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

---

One of two public output functions.

Returns `xspn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

### **sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc () [inline], [inherited]
Moving backwards in the input stream.
```

#### **Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream< char_type, traits_type >::sentry::sentry()`.

### **sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_filebuf< _CharT, _Traits >::sync () [protected], [virtual], [inherited]
Synchronizes the buffer arrays with the controlled sequences.
```

#### **Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

#### **Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `overflow()`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

### **uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

#### **Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Referenced by `xsgetn()`.

**underflow()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual], [inherited]
Fetches more data from the controlled sequence.
```

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_buf_size`, `_M_destroy_pback()`, `_M_mode`, `_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::ios_base::in`, and `overflow()`.

Referenced by `pbackfail()`.

**xsgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (
    char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_buf_size`, `_M_destroy_pback()`, `_M_mode`, `_M_pback_init`, `_M_reading`, `_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gbump()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::ios_base::in`, `overflow()`, `std::basic_streambuf<_CharT, _Traits>::setg()`, and `std::basic_streambuf<char_type, traits_type>::xsgetn()`.



**xspn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic\_filebuf< _CharT, _Traits >::xspn (
    const char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

|                           |                                        |
|---------------------------|----------------------------------------|
| <a href="#">_↔<br/>_s</a> | A buffer area.                         |
| <a href="#">_↔<br/>_n</a> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf](#)< `_CharT`, `_Traits` >.

References `_M_buf_size`, `_M_mode`, `_M_reading`, `_M_set_buffer()`, [std::ios\\_base::app](#), [std::basic\\_streambuf](#)< `_CharT`, `_Traits` >::`eptr()`, [std::ios\\_base::out](#), [std::basic\\_streambuf](#)< `_CharT`, `_Traits` >::`pbase()`, [std::basic\\_streambuf](#)< `_CharT`, `_Traits` >::`pptr()`, and [std::basic\\_streambuf](#)< `char_type`, `traits_type` >::`xspn()`.

**5.904.4 Member Data Documentation****`_M_buf`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_filebuf< _CharT, _Traits >::_M_buf [protected], [inherited]
```

Pointer to the beginning of internal buffer.

Referenced by [basic\\_filebuf\(\)](#), and [setbuf\(\)](#).

**`_M_buf_locale`**

```
template<typename _CharT, typename _Traits>
locale std::basic\_streambuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Referenced by [std::basic\\_filebuf](#)< `_CharT`, `_Traits` >::`basic_filebuf()`.

**`_M_buf_size`**

```
template<typename _CharT, typename _Traits>
size_t std::basic\_filebuf< _CharT, _Traits >::_M_buf_size [protected], [inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf](#)< `_CharT`, `_Traits` >::`stdio_filebuf()`, [\\_\\_gnu\\_cxx::stdio\\_filebuf](#)< `_CharT`, `_Traits` >::`overflow()`, [setbuf\(\)](#), [underflow\(\)](#), [xsgetn\(\)](#), and [xspn\(\)](#).

**`_M_ext_buf`**

```
template<typename _CharT, typename _Traits>
char* std::basic\_filebuf< _CharT, _Traits >::_M_ext_buf [protected], [inherited]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Referenced by [basic\\_filebuf\(\)](#), and [imbue\(\)](#).

### `_M_ext_buf_size`

```
template<typename _CharT, typename _Traits>
streamsize std::basic\_filebuf<\_CharT, \_Traits>::\_M\_ext\_buf\_size [protected], [inherited]
```

Size of buffer held by `_M_ext_buf`.

Referenced by [basic\\_filebuf\(\)](#).

### `_M_ext_next`

```
template<typename _CharT, typename _Traits>
const char* std::basic\_filebuf<\_CharT, \_Traits>::\_M\_ext\_next [protected], [inherited]
```

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Referenced by [basic\\_filebuf\(\)](#), and [imbue\(\)](#).

### `_M_in_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf<\_CharT, \_Traits>::\_M\_in\_beg [protected], [inherited]
```

Start of get area.

### `_M_in_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf<\_CharT, \_Traits>::\_M\_in\_cur [protected], [inherited]
```

Current read area.

### `_M_in_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf<\_CharT, \_Traits>::\_M\_in\_end [protected], [inherited]
```

End of get area.

### `_M_mode`

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic\_filebuf<\_CharT, \_Traits>::\_M\_mode [protected], [inherited]
```

Place to stash in || out || in | out settings for current filebuf.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [showmanyc\(\)](#), [underflow\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

### `_M_out_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf<\_CharT, \_Traits>::\_M\_out\_beg [protected], [inherited]
```

Start of put area.

### `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf<\_CharT, \_Traits>::\_M\_out\_cur [protected], [inherited]
```

Current put area.

### **`_M_out_end`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
End of put area.
```

### **`_M_pback`**

```
template<typename _CharT, typename _Traits>
char_type std::basic\_filebuf< _CharT, _Traits >::_M_pback [protected], [inherited]
Necessary bits for putback buffer management.
```

#### **Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

### **`_M_pback_cur_save`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_filebuf< _CharT, _Traits >::_M_pback_cur_save [protected], [inherited]
Necessary bits for putback buffer management.
```

#### **Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

### **`_M_pback_end_save`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic\_filebuf< _CharT, _Traits >::_M_pback_end_save [protected], [inherited]
Necessary bits for putback buffer management.
```

#### **Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

### **`_M_pback_init`**

```
template<typename _CharT, typename _Traits>
bool std::basic\_filebuf< _CharT, _Traits >::_M_pback_init [protected], [inherited]
Necessary bits for putback buffer management.
```

#### **Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#), [close\(\)](#), [pbackfail\(\)](#), and [xsgetn\(\)](#).

### **`_M_reading`**

```
template<typename _CharT, typename _Traits>
bool std::basic\_filebuf< _CharT, _Traits >::_M_reading [protected], [inherited]
_M_reading == false && _M_writing == false for uncommitted mode; _M_reading == true for read mode; _M_writing
== true for write mode;
```

NB: `_M_reading == true` && `_M_writing == true` is unused.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [seekoff\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

The documentation for this class was generated from the following file:

- [stdio\\_filebuf.h](#)

## 5.905 `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>` Class Template Reference

```
#include <stdio_sync_filebuf.h>
```

Inheritance diagram for `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::off_type` **off\_type**
- typedef `traits_type::pos_type` **pos\_type**
- typedef `_Traits` **traits\_type**

### Public Member Functions

- **stdio\_sync\_filebuf** (`std::__c_file *__f`)
- **stdio\_sync\_filebuf** ([stdio\\_sync\\_filebuf](#) &&\_\_fb) noexcept
- `std::__c_file * file ()`
- locale [getloc](#) () const
- streamsize [in\\_avail](#) ()
- **stdio\_sync\_filebuf & operator=** ([stdio\\_sync\\_filebuf](#) &&\_\_fb) noexcept
- locale [pubimbue](#) (const locale &\_\_loc)
- `int_type sbumpc ()`
- `int_type sgetc ()`
- streamsize [sgetn](#) (char\_type \*\_\_s, streamsize \_\_n)
- `int_type snextc ()`
- `int_type sputbackc (char_type __c)`
- `int_type sputc (char_type __c)`
- streamsize [sputn](#) (const char\_type \*\_\_s, streamsize \_\_n)

- int\_type [sungetc](#) ()
- void [swap](#) ([stdio\\_sync\\_filebuf](#) &\_\_fb)
- [basic\\_streambuf](#) \* [pubsetbuf](#) (char\_type \* \_\_s, streamsize \_\_n)
- pos\_type [pubseekoff](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- pos\_type [pubseekpos](#) (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- int [pubsync](#) ()

### Protected Member Functions

- void [\\_\\_safe\\_gbump](#) (streamsize \_\_n)
- void [\\_\\_safe\\_pbump](#) (streamsize \_\_n)
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const locale &\_\_loc)
- virtual int\_type [overflow](#) (int\_type \_\_c=traits\_type::eof())
- virtual int\_type [pbackfail](#) (int\_type \_\_c=traits\_type::eof())
- void [pbump](#) (int \_\_n)
- virtual [std::streampos seekoff](#) ([std::streamoff](#) \_\_off, [std::ios\\_base::seekdir](#) \_\_dir, [std::ios\\_base::openmode](#)=[std::ios\\_base::in](#)|[std::ios\\_base::out](#))
- virtual pos\_type [seekoff](#) (off\_type, ios\_base::seekdir, ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual [std::streampos seekpos](#) ([std::streampos](#) \_\_pos, [std::ios\\_base::openmode](#) \_\_mode=[std::ios\\_base::in](#)|[std::ios\\_base::out](#))
- virtual pos\_type [seekpos](#) (pos\_type, ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual [basic\\_streambuf](#)< char\_type, \_Traits > \* [setbuf](#) (char\_type \*, streamsize)
- void [setg](#) (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
- void [setp](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
- virtual streamsize [showmanyc](#) ()
- void [swap](#) ([basic\\_streambuf](#) &\_\_sb)
- virtual int [sync](#) ()
- int\_type [syncgetc](#) ()
- [stdio\\_sync\\_filebuf](#)< char >::int\_type [syncgetc](#) ()
- [stdio\\_sync\\_filebuf](#)< wchar\_t >::int\_type [syncgetc](#) ()
- int\_type [syncputc](#) (int\_type \_\_c)
- [stdio\\_sync\\_filebuf](#)< char >::int\_type [syncputc](#) (int\_type \_\_c)
- [stdio\\_sync\\_filebuf](#)< wchar\_t >::int\_type [syncputc](#) (int\_type \_\_c)
- int\_type [syncungetc](#) (int\_type \_\_c)
- [stdio\\_sync\\_filebuf](#)< char >::int\_type [syncungetc](#) (int\_type \_\_c)
- [stdio\\_sync\\_filebuf](#)< wchar\_t >::int\_type [syncungetc](#) (int\_type \_\_c)
- virtual int\_type [uflow](#) ()
- virtual int\_type [underflow](#) ()
- [std::streamsize xsgetn](#) (char \* \_\_s, [std::streamsize](#) \_\_n)
- virtual [std::streamsize xsgetn](#) (char\_type \* \_\_s, [std::streamsize](#) \_\_n)
- [std::streamsize xsgetn](#) (wchar\_t \* \_\_s, [std::streamsize](#) \_\_n)
- virtual streamsize [xsgetn](#) (char\_type \* \_\_s, streamsize \_\_n)
- [std::streamsize xsputn](#) (const char \* \_\_s, [std::streamsize](#) \_\_n)
- virtual [std::streamsize xsputn](#) (const char\_type \* \_\_s, [std::streamsize](#) \_\_n)
- [std::streamsize xsputn](#) (const wchar\_t \* \_\_s, [std::streamsize](#) \_\_n)
- virtual streamsize [xsputn](#) (const char\_type \* \_\_s, streamsize \_\_n)
- char\_type \* [eback](#) () const
- char\_type \* [gptr](#) () const

- `char_type * egptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * ep_ptr () const`

### Protected Attributes

- locale `_M_buf_locale`
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`

#### 5.905.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>
```

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

#### 5.905.2 Member Function Documentation

##### `eback()`

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

##### `egptr()`

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence

- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::xsgetn()`, and `xsgetn()`.

### **epptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::epptr () const [inline], [protected], [inherited]
Access to the put area.
```

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::xsputn()`, and `xsputn()`.

### **file()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
std::_c_file * __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file () [inline]
```

#### **Returns**

The underlying `FILE*`.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

### **gbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::gbump (
    int __n) [inline], [protected], [inherited]
```

Moving the read position.

#### **Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Referenced by `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::getloc () const [inline], [inherited]
Locale access.
```

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

**gptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::gptr () const [inline], [protected], [inherited]
Access to the get area.
```

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::xsgetn()`, and `xsgetn()`.

**imbue()**

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf<_CharT, _Traits>::imbue (
    const locale & __loc) [inline], [protected], [virtual], [inherited]
```

Changes translations.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, and `std::basic_filebuf<wchar_t>`.



**in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**overflow()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::overflow (
    int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

**pbackfail()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::pbackfail (
    int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::sync\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
    int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#).

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::sync\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::xsputn\(\)](#).

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__off</code>  | Offset.                       |
| <code>__way</code>  | Value for ios_base::seekdir.  |
| <code>__mode</code> | Value for ios_base::openmode. |

Calls virtual seekoff function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__sp</code>   | Position                      |
| <code>__mode</code> | Value for ios_base::openmode. |

Calls virtual seekpos function.

**pubsetbuf()**

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
```

```
char_type * __s,
streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

### **pubsync()**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf<_CharT, _Traits>::pubsync () [inline], [inherited]
```

Calls virtual sync function.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

### **sbumpc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sbumpc () [inline], [inherited]
```

Getting the next character.

#### **Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`.

### **seekoff()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf<_CharT, _Traits>::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### **Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_stringbuf<char>`, `std::basic_stringbuf<char>`, `std::basic_stringbuf<wchar_t>`, and `std::basic_stringbuf<wchar_t>`.

### **seekpos()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf<_CharT, _Traits>::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<char_type, traits_type, _Alloc>`, `std::basic_filebuf<wchar_t>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_stringbuf<char>`, `std::basic_stringbuf<char>`, `std::basic_stringbuf<wchar_t>`, and `std::basic_stringbuf<wchar_t>`.

**setbuf()**

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
    char_type * ,
    streamsize ) [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

**Note**

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_stringbuf<char>`, and `std::basic_stringbuf<char>`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
```

```
char_type * __pbeg,
char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

#### Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::wbuffer\\_convert\(\)](#).

#### **sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc () [inline], [inherited]
```

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#).

#### **sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

#### Parameters

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

#### **showmanyc()**

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf<_CharT, _Traits>::showmanyc () [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_stringbuf<char>`, `std::basic_stringbuf<char_type, traits_type>`, `std::basic_stringbuf<wchar_t>`, and `std::basic_stringbuf<wchar_t>`.

### snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::snextc () [inline], [inherited]
```

Getting the next character.

### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<char_type, traits_type>::ignore()`, `std::basic_istream<char_type, traits_type>::operator>>()`, and `std::basic_istream<char_type, traits_type>::putback()`.

### sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (
    char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

### Returns

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
    char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.



**Parameters**

|                                    |                        |
|------------------------------------|------------------------|
| <code>↵</code><br><code>__c</code> | A character to output. |
|------------------------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↵  
__c)`.

Referenced by [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::overflow\(\)](#).

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
    const char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                                    |                     |
|------------------------------------|---------------------|
| <code>↵</code><br><code>__s</code> | A buffer read area. |
| <code>↵</code><br><code>__n</code> | A count.            |

One of two public output functions.

Returns `xsputn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by [std::basic\\_istream<char\\_type, traits\\_type>::sentry::sentry\(\)](#).

**sync()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::sync () [inline], [protected],
[virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

**uflow()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::uflow () [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

**underflow()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::underflow () [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

**xsggetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xsggetn (
    char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

## Parameters

|                                  |                                         |
|----------------------------------|-----------------------------------------|
| <a href="#"><code>__s</code></a> | A buffer area.                          |
| <a href="#"><code>__n</code></a> | Maximum number of characters to assign. |

## Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), and [std::basic\\_filebuf<wchar\\_t>](#).  
References [egptr\(\)](#), [gptr\(\)](#), [std::min\(\)](#), and [uflow\(\)](#).

**xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xsputn (
    const char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

## Parameters

|                                  |                                        |
|----------------------------------|----------------------------------------|
| <a href="#"><code>__s</code></a> | A buffer area.                         |
| <a href="#"><code>__n</code></a> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), and [std::basic\\_filebuf<wchar\\_t>](#).  
References [epptr\(\)](#), [std::min\(\)](#), [overflow\(\)](#), and [pptr\(\)](#).

**5.905.3 Member Data Documentation****`_M_buf_locale`**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::basic\\_filebuf\(\)](#).

**\_M\_in\_beg**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
Start of get area.
```

**\_M\_in\_cur**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
Current read area.
```

**\_M\_in\_end**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
End of get area.
```

**\_M\_out\_beg**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
Start of put area.
```

**\_M\_out\_cur**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
Current put area.
```

**\_M\_out\_end**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
End of put area.
```

The documentation for this class was generated from the following file:

- [stdio\\_sync\\_filebuf.h](#)

**5.906 std::chrono::steady\_clock Struct Reference**

```
#include <chrono.h>
```

**Public Types**

- typedef [chrono::nanoseconds](#) **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef [chrono::time\\_point](#)< [steady\\_clock](#), [duration](#) > **time\_point**

**Static Public Member Functions**

- static [time\\_point](#) **now** () noexcept

## Static Public Attributes

- static constexpr bool **is\_steady**

### 5.906.1 Detailed Description

Monotonic clock.

Time returned has the property of only increasing at a uniform rate.

The documentation for this struct was generated from the following file:

- [chrono.h](#)

## 5.907 `__gnu_pbds::detail::stored_data<_Tv, _Th, Store_Hash >` Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, _Th, Store_Hash >`:



## Public Types

- typedef `_Th` **hash\_type**
- typedef `_Tv` **value\_type**

## Public Attributes

- `hash_type` **m\_hash**
- `value_type` **m\_value**

### 5.907.1 Detailed Description

```
template<typename _Tv, typename _Th, bool Store_Hash>
struct __gnu_pbds::detail::stored_data<_Tv, _Th, Store_Hash >
```

Primary template for representation of stored data. Two types of data can be stored: value and hash.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

**5.908 `__gnu_pbds::detail::stored_data<_Tv, _Th, false >` Struct Template Reference**

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, _Th, false >`:

**Public Types**

- `typedef _Th hash_type`
- `typedef _Tv value_type`

**Public Attributes**

- `hash_type m_hash`
- `value_type m_value`

**5.908.1 Detailed Description**

```
template<typename _Tv, typename _Th>
struct __gnu_pbds::detail::stored_data<_Tv, _Th, false >
```

Specialization for representation of stored data of just value type.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

**5.909 `__gnu_pbds::detail::stored_hash<_Th >` Struct Template Reference**

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::stored_hash<_Th>`:



### Public Types

- typedef `_Th` `hash_type`

### Public Attributes

- `hash_type` `m_hash`

#### 5.909.1 Detailed Description

```
template<typename _Th>
struct __gnu_pbds::detail::stored_hash<_Th>
```

Stored hash.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 5.910 `__gnu_pbds::detail::stored_value<_Tv>` Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::stored\_value<\_Tv>:



### Public Types

- typedef `_Tv` **value\_type**

### Public Attributes

- `value_type` **m\_value**

#### 5.910.1 Detailed Description

```
template<typename _Tv>
struct __gnu_pbds::detail::stored_value<_Tv>
```

Stored value.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

### 5.911 \_\_gnu\_pbds::string\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```



Inheritance diagram for `__gnu_pbds::string_tag`:



#### 5.911.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 5.912 `std::student_t_distribution<_RealType>` Class Template Reference

```
#include <random>
```

##### Classes

- struct [param\\_type](#)

##### Public Types

- typedef `_RealType` [result\\_type](#)

##### Public Member Functions

- **student\_t\_distribution** (`_RealType __n`)
- **student\_t\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator`>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator`>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)

- `result_type max () const`
- `result_type min () const`
- `_RealType n () const`
- `template<typename _UniformRandomNumberGenerator>  
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator>  
result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::student_t_distribution< _RealType1 > &__x)`
- `bool operator== (const student_t_distribution &__d1, const student_t_distribution &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::student_t_distribution< _RealType1 > &__x)`

### 5.912.1 Detailed Description

**template<typename \_RealType = double>**  
**class std::student\_t\_distribution<\_RealType>**

A student\_t\_distribution random number distribution.

The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Since

C++11

### 5.912.2 Member Typedef Documentation

#### result\_type

`template<typename _RealType = double>`  
`typedef _RealType std::student_t_distribution<_RealType>::result_type`  
 The type of the range of the distribution.

### 5.912.3 Member Function Documentation

#### max()

`template<typename _RealType = double>`  
`result_type std::student_t_distribution<_RealType>::max () const [inline]`  
 Returns the least upper bound value of the distribution.

#### min()

`template<typename _RealType = double>`  
`result_type std::student_t_distribution<_RealType>::min () const [inline]`  
 Returns the greatest lower bound value of the distribution.

**operator()()**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::student_t_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

**param() [1/2]**

```
template<typename _RealType = double>
param_type std::student_t_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

**5.912.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::student_t_distribution< _RealType1 > & __x) [friend]
```

Inserts a student\_t\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <code>__os</code> | An output stream.                                    |
| <code>__x</code>  | A student_t_distribution random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
    const student_t_distribution< _RealType > & __d1,
    const student_t_distribution< _RealType > & __d2) [friend]
```

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::student_t_distribution< _RealType1 > & __x) [friend]
```

Extracts a student\_t\_distribution random number distribution \_\_x from the input stream \_\_is.

**Parameters**

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| <code>__is</code> | An input stream.                                         |
| <code>__x</code>  | A student_t_distribution random number generator engine. |

**Returns**

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.913 std::sub\_match<\_Bilter > Class Template Reference**

```
#include <regex>
```

**Public Types**

- typedef \_\_iter\_traits::difference\_type **difference\_type**
- typedef \_Bilter **iterator**
- typedef basic\_string< value\_type > **string\_type**
- typedef \_\_iter\_traits::value\_type **value\_type**

**Public Member Functions**

- int [compare](#) (const [sub\\_match](#) &\_\_s) const
- difference\_type [length](#) () const noexcept
- [operator string\\_type](#) () const
- [string\\_type](#) [str](#) () const
- void [swap](#) ([sub\\_match](#) &\_\_s) noexcept(\_\_is\_nothrow\_swappable< \_Bilter >::value)
- int [compare](#) (const [string\\_type](#) &\_\_s) const
- int [compare](#) (const value\_type \*\_\_s) const

## Public Attributes

- iterator **first**
- bool **matched**
- iterator **second**

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _Bilter>`  
`bool operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bilter>`  
`auto operator<=> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bilter >::value)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Alloc > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Bi_iter>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Bi_iter>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>`  
`basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`

### 5.913.1 Detailed Description

`template<typename _Bilter>`  
`class std::sub_match< _Bilter >`

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with `std::basic_string` objects of the same character type as the pattern matched by the regular expression.

A `sub_match<Iter>` has a public base class of type `pair<Iter, Iter>`, so inherits `pair`'s data members named `first` and `second`. The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Since

C++11

## 5.913.2 Member Function Documentation

**compare()** [1/3]

```
template<typename _BiIter>
int std::sub_match<_BiIter >::compare (
    const string_type & __s) const [inline]
```

Compares this sub\_match to a string.

**Parameters**

|                                                                                          |                                        |
|------------------------------------------------------------------------------------------|----------------------------------------|
| <br>__s | A string to compare to this sub_match. |
|------------------------------------------------------------------------------------------|----------------------------------------|

**Return values**

|                 |                                                |
|-----------------|------------------------------------------------|
| <i>negative</i> | This matched sequence will collate before __s. |
| <i>zero</i>     | This matched sequence is equivalent to __s.    |
| <i>positive</i> | This matched sequence will collate after __s.  |

**compare()** [2/3]

```
template<typename _BiIter>
int std::sub_match<_BiIter >::compare (
    const sub_match<_BiIter > & __s) const [inline]
```

Compares this and another matched sequence.

**Parameters**

|                                                                                            |                                                  |
|--------------------------------------------------------------------------------------------|--------------------------------------------------|
| <br>__s | Another matched sequence to compare to this one. |
|--------------------------------------------------------------------------------------------|--------------------------------------------------|

**Return values**

|                 |                                                |
|-----------------|------------------------------------------------|
| <i>negative</i> | This matched sequence will collate before __s. |
| <i>zero</i>     | This matched sequence is equivalent to __s.    |
| <i>positive</i> | This matched sequence will collate after __s.  |

Referenced by [std::sub\\_match< const char \\* >::operator==\(, and std::sub\\_match< const char \\* >::operator==\(.](#)

**compare()** [3/3]

```
template<typename _BiIter>
int std::sub_match<_BiIter >::compare (
    const value_type * __s) const [inline]
```

Compares this sub\_match to a string.

**Parameters**

|                                                                                            |                                        |
|--------------------------------------------------------------------------------------------|----------------------------------------|
| <br>__s | A string to compare to this sub_match. |
|--------------------------------------------------------------------------------------------|----------------------------------------|

**Return values**

|                 |                                                |
|-----------------|------------------------------------------------|
| <i>negative</i> | This matched sequence will collate before __s. |
| <i>zero</i>     | This matched sequence is equivalent to __s.    |
| <i>positive</i> | This matched sequence will collate after __s.  |

**length()**

```
template<typename _BiIter>
difference_type std::sub_match< _BiIter >::length () const [inline], [noexcept]
Gets the length of the matching sequence.
```

**operator string\_type()**

```
template<typename _BiIter>
string_type std::sub_match< _BiIter >::operator string_type () const [inline]
Gets the matching sequence as a string.
```

**Returns**

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the str() member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

**str()**

```
template<typename _BiIter>
string_type std::sub_match< _BiIter >::str () const [inline]
Gets the matching sequence as a string.
```

**Returns**

the matching sequence as a string.

Referenced by [std::sub\\_match< const char \\* >::operator string\\_type\(\)](#).

**swap()**

```
template<typename _BiIter>
void std::sub_match< _BiIter >::swap (
    sub_match< _BiIter > & __s) [inline], [noexcept]
```

Swap the values of two sub\_match objects.

The documentation for this class was generated from the following file:

- [regex.h](#)

**5.914 std::ranges::subrange< \_It, \_Sent, \_Kind > Class Template Reference**

```
#include <ranges_util.h>
```

Inheritance diagram for std::ranges::subrange<\_It, \_Sent, \_Kind>:



### Public Member Functions

- constexpr **subrange** (\_\_detail::\_\_convertible\_to\_non\_slicing<\_It> auto \_\_i, \_Sent \_\_s) noexcept(is\_nothrow\_constructible\_v<\_It, decltype(\_\_i)> &&is\_nothrow\_constructible\_v<\_Sent, \_Sent &>)
- constexpr **subrange** (\_\_detail::\_\_convertible\_to\_non\_slicing<\_It> auto \_\_i, \_Sent \_\_s, \_\_size\_type \_\_n) noexcept(is\_nothrow\_constructible\_v<\_It, decltype(\_\_i)> &&is\_nothrow\_constructible\_v<\_Sent, \_Sent &>)
- template<\_\_detail::\_\_different\_from<subrange> \_Rng>  
requires borrowed\_range<\_Rng> && \_\_detail::\_\_convertible\_to\_non\_slicing<iterator\_t<\_Rng>, \_It> && convertible\_to<sentinel\_t<\_Rng>, \_Sent>  
constexpr **subrange** (\_Rng &&\_\_r) noexcept(noexcept(subrange(\_\_r, ranges::size(\_\_r))) \_S\_store\_size &&sized\_range<\_Rng>)
- template<\_\_detail::\_\_different\_from<subrange> \_Rng>  
requires borrowed\_range<\_Rng> && \_\_detail::\_\_convertible\_to\_non\_slicing<iterator\_t<\_Rng>, \_It> && convertible\_to<sentinel\_t<\_Rng>, \_Sent> (!\_S\_store\_size)  
constexpr **subrange** (\_Rng &&\_\_r) noexcept(noexcept(subrange(ranges::begin(\_\_r), ranges::end(\_\_r))))
- template<borrowed\_range \_Rng>  
requires \_\_detail::\_\_convertible\_to\_non\_slicing<iterator\_t<\_Rng>, \_It> && convertible\_to<sentinel\_t<\_Rng>, \_Sent> (\_Kind == subrange\_kind::sized)  
constexpr **subrange** (\_Rng &&\_\_r, \_\_size\_type \_\_n) noexcept(noexcept(subrange(ranges::begin(\_\_r), ranges::end(\_\_r), \_\_n)))
- constexpr subrange & **advance** (iter\_difference\_t<\_It> \_\_n)
- constexpr decltype(auto) **back** ()
- constexpr decltype(auto) **back** () const
- constexpr \_It **begin** ()
- constexpr \_It **begin** () const
- constexpr auto **data** () const noexcept(noexcept(ranges::begin(\_M\_derived())))
- constexpr auto **data** () noexcept(noexcept(ranges::begin(\_M\_derived())))
- constexpr bool **empty** () const
- constexpr bool **empty** () noexcept(noexcept(\_S\_empty(\_M\_derived())))
- constexpr \_Sent **end** () const
- constexpr decltype(auto) **front** ()
- constexpr decltype(auto) **front** () const



- constexpr [subrange](#) **next** (iter\_difference\_t< \_It > \_\_n=1) &&
- constexpr [subrange](#) **next** (iter\_difference\_t< \_It > \_\_n=1) const &
- template<\_\_detail::\_\_different\_from< [subrange](#) > \_PairLike>  
requires \_\_detail::\_\_pair\_like\_convertible\_from<\_PairLike, const \_It&, const \_Sent&>  
constexpr **operator** \_PairLike () const
- constexpr **operator** bool () const noexcept(noexcept(ranges::empty(\_M\_derived())))
- constexpr **operator** bool () noexcept(noexcept(ranges::empty(\_M\_derived())))
- template<random\_access\_range \_Range = \_Derived>  
constexpr decltype(auto) **operator**[] (range\_difference\_t< \_Range > \_\_n)
- template<random\_access\_range \_Range = const \_Derived>  
constexpr decltype(auto) **operator**[] (range\_difference\_t< \_Range > \_\_n) const
- constexpr [subrange](#) **prev** (iter\_difference\_t< \_It > \_\_n=1) const
- constexpr \_\_size\_type **size** () const
- constexpr auto **size** () noexcept(noexcept(\_S\_size(\_M\_derived())))

## Friends

- struct **views::\_Drop**

### 5.914.1 Detailed Description

```
template<input_or_output_iterator _It, sentinel_for< _It > _Sent = _It, subrange_kind _Kind = sized_sentinel,
_for<_Sent, _It> ? subrange_kind::sized : subrange_kind::unsized>
requires (_Kind == subrange_kind::sized || !sized_sentinel_for<_Sent, _It>)
class std::ranges::subrange< _It, _Sent, _Kind >
```

The ranges::subrange class template.

The documentation for this class was generated from the following file:

- [ranges\\_util.h](#)

## 5.915 std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r > Class Template Reference

```
#include <random>
```

### Public Types

- typedef \_UIntType [result\\_type](#)

### Public Member Functions

- template<typename \_Sseq, typename = \_If\_seed\_seq<\_Sseq>>  
[subtract\\_with\\_carry\\_engine](#) (\_Sseq &\_\_q)
- [subtract\\_with\\_carry\\_engine](#) ([result\\_type](#) \_\_sd)
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator](#)() ()
- template<typename \_Sseq>  
\_If\_seed\_seq< \_Sseq > [seed](#) (\_Sseq &\_\_q)
- template<typename \_Sseq>  
auto [seed](#) (\_Sseq &\_\_q) -> \_If\_seed\_seq< \_Sseq >
- void [seed](#) ([result\\_type](#) \_\_sd=0u)

### Static Public Member Functions

- static constexpr [result\\_type](#) max ()
- static constexpr [result\\_type](#) min ()

### Static Public Attributes

- static constexpr uint\_least32\_t **default\_seed**
- static constexpr size\_t **long\_lag**
- static constexpr size\_t **short\_lag**
- static constexpr size\_t **word\_size**

### Friends

- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_s1, size\_t \_\_r1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::subtract\\_with\\_carry\\_engine](#)< \_UIntType1, \_\_w1, \_\_s1, \_\_r1 > &\_\_x)
- bool [operator==](#) (const [subtract\\_with\\_carry\\_engine](#) &\_\_lhs, const [subtract\\_with\\_carry\\_engine](#) &\_\_rhs)
- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_s1, size\_t \_\_r1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::subtract\\_with\\_carry\\_engine](#)< \_UIntType1, \_\_w1, \_\_s1, \_\_r1 > &\_\_x)

#### 5.915.1 Detailed Description

**template<typename \_UIntType, size\_t \_\_w, size\_t \_\_s, size\_t \_\_r>**  
**class std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >**

The Marsaglia-Zaman generator.

This is a model of a Generalized Fibonacci discrete random number generator, sometimes referred to as the SWC generator.

A discrete random number generator that produces pseudorandom numbers using:

$$x_i \leftarrow (x_{i-s} - x_{i-r} - carry_{i-1}) \bmod m$$

The size of the state is  $r$  and the maximum period of the generator is  $(m^r - m^s - 1)$ .

Since

C++11

#### 5.915.2 Member Typedef Documentation

##### result\_type

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
typedef _UIntType std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r >::result_type
```

The type of the generated random value.

#### 5.915.3 Constructor & Destructor Documentation

##### subtract\_with\_carry\_engine() [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
    result\_type __sd) [inline], [explicit]
```

Constructs an explicitly seeded [subtract\\_with\\_carry\\_engine](#) random number generator.

**subtract\_with\_carry\_engine()** [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
    _Sseq & __q) [inline], [explicit]
```

Constructs a `subtract_with_carry_engine` random number engine seeded from the seed sequence `__q`.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

**5.915.4 Member Function Documentation****discard()**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::discard (
    unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

**max()**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
static constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::max ()
[inline], [static], [constexpr]
```

Gets the inclusive maximum value of the range of random integers returned by this generator.

**min()**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
static constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::min ()
[inline], [static], [constexpr]
```

Gets the inclusive minimum value of the range of random integers returned by this generator.

**operator()()**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type std::subtract_with_carry_engine<
    _UIntType, __w, __s, __r >::operator() ()
```

Gets the next random number in the sequence.

**seed()** [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
    _Sseq & __q)
```

Seeds the initial state  $x_0$  of the % `subtract_with_carry_engine` random number generator.

**seed()** [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
    result_type __sd = 0u)
```

Seeds the initial state  $x_0$  of the random number generator.

N1688[4.19] modifies this as follows. If `__value == 0`, sets value to 19780503. In any case, with a linear congruential generator  $lcg(i)$  having parameters  $m_{lcg} = 2147483563$ ,  $a_{lcg} = 40014$ ,  $c_{lcg} = 0$ , and  $lcg(0) = value$ , sets  $x_{-r} \dots x_{-1}$  to  $lcg(1) \bmod m \dots lcg(r) \bmod m$  respectively. If  $x_{-1} = 0$  set carry to 1, otherwise sets carry to 0.

Referenced by `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::subtract_with_carry_engine()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::subtract_with_carry_engine()`.

**5.915.5 Friends And Related Symbol Documentation****operator<<**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x) [friend]
```

Inserts the current state of a % `subtract_with_carry_engine` random number generator engine `__x` into the output stream `__os`.

**Parameters**

|                   |                                                                             |
|-------------------|-----------------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                           |
| <code>__x</code>  | A % <code>subtract_with_carry_engine</code> random number generator engine. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool operator== (
    const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs,
    const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs) [friend]
```

Compares two % `subtract_with_carry_engine` random number generator objects of the same type for equality.

**Parameters**

|                    |                                                                                   |
|--------------------|-----------------------------------------------------------------------------------|
| <code>__lhs</code> | A % <code>subtract_with_carry_engine</code> random number generator object.       |
| <code>__rhs</code> | Another % <code>subtract_with_carry_engine</code> random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

**operator>>**

```

template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x) [friend]

```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

**Parameters**

|                   |                                                                |
|-------------------|----------------------------------------------------------------|
| <code>__is</code> | An input stream.                                               |
| <code>__x</code>  | A % subtract_with_carry_engine random number generator engine. |

**Returns**

The input stream with the state of \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.916 \_\_gnu\_cxx::subtractive\_rng Class Reference**

```
#include <functional>
```

Inheritance diagram for \_\_gnu\_cxx::subtractive\_rng:

**Public Types**

- typedef unsigned int [argument\\_type](#)
- typedef unsigned int [result\\_type](#)

## Public Member Functions

- [subtractive\\_rng](#) ()
- [subtractive\\_rng](#) (unsigned int \_\_seed)
- void **\_M\_initialize** (unsigned int \_\_seed)
- unsigned int [operator\(\)](#) (unsigned int \_\_limit)

### 5.916.1 Detailed Description

The `subtractive_rng` class is documented on [SGI's site](#). Note that this code assumes that `int` is 32 bits.

### 5.916.2 Member Typedef Documentation

#### `argument_type`

typedef unsigned int [std::unary\\_function](#)< unsigned int, unsigned int >::argument\_type [inherited]  
`argument_type` is the type of the argument

#### `result_type`

typedef unsigned int [std::unary\\_function](#)< unsigned int, unsigned int >::result\_type [inherited]  
`result_type` is the return type

### 5.916.3 Constructor & Destructor Documentation

#### `subtractive_rng()` [1/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng (
    unsigned int __seed) [inline]
```

Ctor allowing you to initialize the seed.

#### `subtractive_rng()` [2/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng () [inline]
```

Default ctor; initializes its state with some number you don't see.

### 5.916.4 Member Function Documentation

#### `operator()()`

```
unsigned int __gnu_cxx::subtractive_rng::operator() (
    unsigned int __limit) [inline]
```

Returns a number less than the argument.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.917 `__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >` Struct Template Reference

```
#include <synth_access_traits.hpp>
```

## Public Types

- typedef `_ATraits` **base\_type**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `type_traits::const_reference` **const\_reference**

- typedef type\_traits::key\_const\_reference **key\_const\_reference**
- typedef Type\_Traits **type\_traits**

#### Public Member Functions

- **synth\_access\_traits** (const base\_type &)
- bool **cmp\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **cmp\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=false) const
- bool **equal\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **equal\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=true) const

#### Static Public Member Functions

- static key\_const\_reference **extract\_key** (const\_reference)

#### 5.917.1 Detailed Description

```
template<typename Type_Traits, bool Set, typename _ATraits>
struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >
```

Synthetic element access traits.

The documentation for this struct was generated from the following file:

- [synth\\_access\\_traits.hpp](#)

### 5.918 std::chrono::system\_clock Struct Reference

```
#include <chrono.h>
```

#### Public Types

- typedef [chrono::nanoseconds](#) **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef [chrono::time\\_point](#)< [system\\_clock](#), [duration](#) > **time\_point**

#### Static Public Member Functions

- static [time\\_point](#) **from\_time\_t** (std::time\_t \_\_t) noexcept
- static [time\\_point](#) **now** () noexcept
- static std::time\_t **to\_time\_t** (const [time\\_point](#) &\_\_t) noexcept

#### Static Public Attributes

- static constexpr bool **is\_steady**

#### 5.918.1 Detailed Description

System clock.

Time returned represents wall time from the system-wide clock.

The documentation for this struct was generated from the following file:

- [chrono.h](#)

## 5.919 std::system\_error Class Reference

```
#include <system_error>
```

Inheritance diagram for std::system\_error:



### Public Member Functions

- **system\_error** (const [system\\_error](#) &)=default
- **system\_error** ([error\\_code](#) \_\_ec, const char \*\_\_what)
- **system\_error** ([error\\_code](#) \_\_ec, const [string](#) &\_\_what)
- **system\_error** ([error\\_code](#) \_\_ec=[error\\_code](#)())
- **system\_error** (int \_\_v, const [error\\_category](#) &\_\_ecat)
- **system\_error** (int \_\_v, const [error\\_category](#) &\_\_ecat, const char \*\_\_what)
- **system\_error** (int \_\_v, const [error\\_category](#) &\_\_ecat, const [string](#) &\_\_what)
- const [error\\_code](#) & **code** () const noexcept
- [system\\_error](#) & **operator=** (const [system\\_error](#) &)=default
- virtual const char \* **what** () const noexcept

### 5.919.1 Detailed Description

An exception type that includes an `error_code` value.

Typically used to report errors from the operating system and other low-level APIs.

Since

C++11

### 5.919.2 Member Function Documentation

#### what()

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), and [std::filesystem::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 5.920 std::chrono::tai\_clock Class Reference

```
#include <chrono>
```



## Public Types

- using **duration**
- using **period**
- using **rep**
- using **time\_point**

## Static Public Member Functions

- `template<typename _Duration>`  
`static tai_time< common_type_t< _Duration, seconds > > from_utc (const utc_time< _Duration > &__t)`
- `static time_point now ()`
- `template<typename _Duration>`  
`static utc_time< common_type_t< _Duration, seconds > > to_utc (const tai_time< _Duration > &__t)`

## Static Public Attributes

- `static constexpr bool is_steady`

### 5.920.1 Detailed Description

A clock that measures International Atomic Time.  
The epoch is 1958-01-01 00:00:00.

Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

### 5.921 `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >` Struct Template Reference

```
#include <memory>
```

Inheritance diagram for `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`:



## Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- `temporary_buffer` (`_ForwardIterator __first`, `_ForwardIterator __last`)
- `~temporary_buffer` ()
- `size_type _M_requested_size` () const
- iterator `begin` ()
- iterator `end` ()
- `size_type size` () const

## Protected Attributes

- struct `std::_Temporary_buffer::Impl` **\_M\_impl**
- `size_type` **\_M\_original\_len**

### 5.921.1 Detailed Description

**template<class \_ForwardIterator, class \_Tp = typename std::iterator\_traits<\_ForwardIterator>::value\_type>**  
**struct \_\_gnu\_cxx::temporary\_buffer<\_ForwardIterator, \_Tp>**

This class provides similar behavior and semantics of the standard functions `get_temporary_buffer()` and `return_temporary_buffer()`, but encapsulated in a type vaguely resembling a standard container.

By default, a `temporary_buffer<Iter>` stores space for objects of whatever type the `Iter` iterator points to. It is constructed from a typical `[first,last)` range, and provides the `begin()`, `end()`, `size()` functions, as well as `requested_size()`. For non-trivial types, copies of `*first` will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like `get_temporary_buffer()`, not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of `[first,last)`, but if `size()` is less than `requested_size()`, then this didn't happen.

### 5.921.2 Constructor & Destructor Documentation

#### `temporary_buffer()`

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>::temporary_buffer (
    _ForwardIterator __first,
    _ForwardIterator __last) [inline]
```

Requests storage large enough to hold a copy of `[first,last)`.

#### `~temporary_buffer()`

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>::~~temporary_buffer () [inline]
```

Destroys objects and frees storage.

### 5.921.3 Member Function Documentation

#### `_M_requested_size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const [inline],
[inherited]
```

Returns the size requested by the constructor; may be >size().

#### `begin()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin () [inline], [inherited]
```

As per Table mumble.

Referenced by `_Temporary_buffer()`.

#### `end()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end () [inline], [inherited]
```

As per Table mumble.

Referenced by `_Temporary_buffer()`.

#### `size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const [inline], [inherited]
```

As per Table mumble.

The documentation for this struct was generated from the following file:

- [ext/memory](#)

### 5.922 `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

```
#include <thin_heap_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`:



#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**

- typedef `__rebind_a::const_pointer` **const\_pointer**
- typedef `__rebind_a::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

### Public Member Functions

- **iterator** `begin` ()
- **const\_iterator** `begin` () const
- void **clear ()**
- bool **empty** () const
- **iterator** `end` ()
- **const\_iterator** `end` () const
- void **erase** (**point\_iterator**)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **join** (**thin\_heap**< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type **max\_size** () const
- void **modify** (**point\_iterator**, const\_reference)
- void **pop** ()
- **point\_iterator** **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, **thin\_heap**< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** (**left\_child\_next\_sibling\_heap**< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference **top** () const

### Protected Types

- typedef `base_type::node` **node**
- typedef `alloc_traits::allocator_type` **node\_allocator**
- typedef `base_type::node_const_pointer` **node\_const\_pointer**
- typedef `_Alloc::size_type` **node\_metadata**
- typedef `base_type::node_pointer` **node\_pointer**
- typedef `std::pair< node_pointer, node_pointer >` **node\_pointer\_pair**

### Protected Member Functions

- **thin\_heap** (const Cmp\_Fn &)
- **thin\_heap** (const **thin\_heap**< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)

- `template<typename It>`  
void **copy\_from\_range** (It, It)
- `node_pointer` **get\_new\_node\_for\_insert** (const\_reference)
- `node_pointer` **prune** (Pred)
- void **swap** ([thin\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static `node_pointer` **parent** (node\_pointer)

### Protected Attributes

- `node_pointer` **m\_p\_root**
- `size_type` **m\_size**

#### 5.922.1 Detailed Description

`template<typename Value_Type, typename Cmp_Fn, typename _Alloc>`  
**class** `__gnu_pbds::detail::thin_heap`< Value\_Type, Cmp\_Fn, \_Alloc >

Thin heap.

See Tarjan and Kaplan.

The documentation for this class was generated from the following file:

- [thin\\_heap.hpp](#)

### 5.923 \_\_gnu\_pbds::thin\_heap\_tag Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for `__gnu_pbds::thin_heap_tag`:



### 5.923.1 Detailed Description

Thin heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.924 std::thread Class Reference

```
#include <thread>
```

### Classes

- class [id](#)

### Public Types

- using [native\\_handle\\_type](#)

### Public Member Functions

- `template<typename _Callable, typename... _Args, typename = _Require<__not_same<_Callable>>>> thread (_Callable &&__f, _Args &&... __args)`
- `thread (const thread &)=delete`
- `thread (thread &&__t) noexcept`
- `void detach ()`
- `id get_id () const noexcept`
- `void join ()`
- `bool joinable () const noexcept`
- `native_handle_type native\_handle ()`
- `thread & operator= (const thread &)=delete`
- `thread & operator= (thread &&__t) noexcept`
- `void swap (thread &__t) noexcept`

### Static Public Member Functions

- `static unsigned int hardware_concurrency () noexcept`

### Related Symbols

(Note that these are not member symbols.)

- `bool operator== (thread::id __x, thread::id __y) noexcept`
- `void swap (thread &__x, thread &__y) noexcept`

### 5.924.1 Detailed Description

A `std::thread` represents a new thread of execution.

The default constructor creates an object that does not own a thread. The `thread(F&&, Args&&...)` constructor invokes a callable in a new thread, and owns that new thread. A `std::thread` that owns a thread is *joinable*. Joining a thread waits for it to finish executing, which happens when the callable running in that thread returns.

A `std::thread` cannot be copied, but can be moved. Moving a joinable object transfers ownership of its thread to another object.

A joinable `std::thread` must be explicitly joined (or detached) before it is destroyed or assigned to. Attempting to destroy a joinable thread will terminate the whole process.

Since

C++11

### 5.924.2 Member Function Documentation

#### native\_handle()

```
native_handle_type std::thread::native_handle () [inline]
```

#### Precondition

thread is joinable

The documentation for this class was generated from the following file:

- [std\\_thread.h](#)

### 5.925 \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond > Class Template Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond >:



#### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **address** (reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*\_\_hint=0)
- void **check** (size\_type \_\_n)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_constructed** (size\_t label)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)

- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up>  
void **destroy** (\_Up \*\_\_p)
- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)
- size\_type **max\_size** () const noexcept

#### Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

#### 5.925.1 Detailed Description

```
template<typename _Tp, typename _Cond>  
class __gnu_cxx::throw_allocator_base<_Tp, _Cond >
```

Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.

Note: Deallocate not allowed to throw.

The documentation for this class was generated from the following file:

- [throw\\_allocator.h](#)

## 5.926 `__gnu_cxx::throw_allocator_limit<_Tp>` Struct Template Reference

```
#include <throw_allocator.h>
```



Inheritance diagram for `__gnu_cxx::throw_allocator_limit< _Tp >`:



## Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#) &) noexcept
- template<typename \_Tp1>  
  **throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#)< \_Tp1 > &) noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **address** (reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \* \_\_hint=0)
- void **check** (size\_type \_\_n)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_constructed** (size\_t label)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- void **construct** (\_Up \* \_\_p, \_Args &&... \_\_args)

- void **deallocate** (pointer `__p`, `size_type __n`)
- void **destroy** (`_Up *__p`)
- void **erase** (`void *p`, `size_t size`)
- void **erase\_construct** (`void *p`)
- void **insert** (`void *p`, `size_t size`)
- void **insert\_construct** (`void *p`)
- `size_type max_size` () const noexcept
- `throw_allocator_limit` & **operator=** (const `throw_allocator_limit` &)=default

#### Static Public Member Functions

- static void **check** ()
- static `size_t` & **count** ()
- static `size_t` **get\_label** ()
- static `size_t` & **limit** ()
- static void **set\_label** (`size_t l`)
- static void **set\_limit** (const `size_t __l`)
- static void **throw\_conditionally** ()

#### 5.926.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_limit<_Tp>
```

Allocator throwing via limit condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 5.927 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::throw_allocator_random< _Tp >`:



## Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **throw\_allocator\_random** (const [throw\\_allocator\\_random](#) &) noexcept
- template<typename \_Tp1>  
  **throw\_allocator\_random** (const [throw\\_allocator\\_random](#)< \_Tp1 > &) noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **address** (reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*\_\_hint=0)
- void **check** (size\_type \_\_n)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)
- void **check\_constructed** (size\_t label)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)

- void **destroy** (`_Up *`\_\_p)
- void **erase** (`void *`p, `size_t` size)
- void **erase\_construct** (`void *`p)
- void **insert** (`void *`p, `size_t` size)
- void **insert\_construct** (`void *`p)
- `size_type` **max\_size** () const noexcept
- [throw\\_allocator\\_random](#) & **operator=** (const [throw\\_allocator\\_random](#) &)=default
- void **seed** (unsigned long \_\_s)

### Static Public Member Functions

- static void **check** ()
- static `size_t` **get\_label** ()
- static void **set\_label** (`size_t` l)
- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

#### 5.927.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_random<_Tp>
```

Allocator throwing via random condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.928 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference

```
#include <throw_allocator.h>
```

### Public Types

- typedef `_Cond` **condition\_type**

### Public Member Functions

- **throw\_value\_base** (const `std::size_t` \_\_i)
- **throw\_value\_base** (const [throw\\_value\\_base](#) &\_\_v)
- **throw\_value\_base** ([throw\\_value\\_base](#) &&)=default
- [throw\\_value\\_base](#) & **operator++** ()
- [throw\\_value\\_base](#) & **operator=** (const [throw\\_value\\_base](#) &\_\_v)
- [throw\\_value\\_base](#) & **operator=** ([throw\\_value\\_base](#) &&)=default

### Public Attributes

- `std::size_t` **\_M\_i**

#### 5.928.1 Detailed Description

```
template<typename _Cond>
struct __gnu_cxx::throw_value_base<_Cond>
```

Class with exception generation control. Intended to be used as a `value_type` in templated code.

Note: Destructor not allowed to throw.

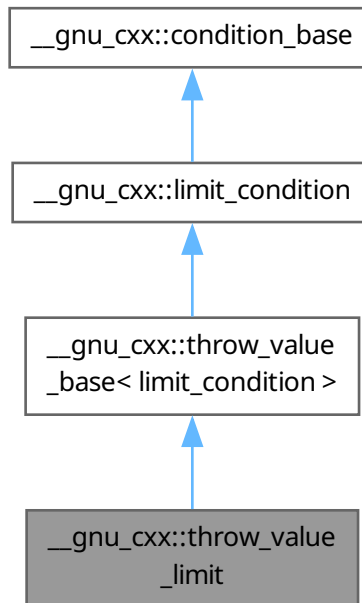
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.929 `__gnu_cxx::throw_value_limit` Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::throw_value_limit`:



### Public Types

- typedef `throw_value_base< limit_condition >` **base\_type**
- typedef `limit_condition` **condition\_type**

### Public Member Functions

- **throw\_value\_limit** (const std::size\_t \_\_i)
- **throw\_value\_limit** (const `throw_value_limit` &\_\_other)
- **throw\_value\_limit** (`throw_value_limit` &&)=default
- `throw_value_base` & **operator++** ()
- `throw_value_base` & **operator++** ()
- `throw_value_limit` & **operator=** (const `throw_value_limit` &\_\_other)
- `throw_value_limit` & **operator=** (`throw_value_limit` &&)=default

### Static Public Member Functions

- static size\_t & **count** ()
- static size\_t & **limit** ()
- static void **set\_limit** (const size\_t \_\_l)
- static void **throw\_conditionally** ()

**Public Attributes**

- `std::size_t _M_i`
- `std::size_t _M_i`

**5.929.1 Detailed Description**

Type throwing via limit condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

**5.930 \_\_gnu\_cxx::throw\_value\_random Struct Reference**

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::throw_value_random`:

**Public Types**

- typedef [throw\\_value\\_base< random\\_condition >](#) **base\_type**
- typedef [random\\_condition](#) **condition\_type**

**Public Member Functions**

- **throw\_value\_random** (const `std::size_t __i`)
- **throw\_value\_random** (const [throw\\_value\\_random](#) &\_\_other)
- **throw\_value\_random** ([throw\\_value\\_random](#) &&)=default
- [throw\\_value\\_base](#) & **operator++** ()

- [throw\\_value\\_base](#) & **operator++** ()
- [throw\\_value\\_random](#) & **operator=** (const [throw\\_value\\_random](#) &\_\_other)
- [throw\\_value\\_random](#) & **operator=** ([throw\\_value\\_random](#) &&)=default
- void **seed** (unsigned long \_\_s)

### Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

### Public Attributes

- std::size\_t **\_M\_i**
- std::size\_t **\_M\_i**

### 5.930.1 Detailed Description

Type throwing via random condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.931 std::time\_base Class Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::time\_base:



### Public Types

- enum **dateorder** {  
  **no\_order** , **dmy** , **mdy** , **ymd** ,  
  **ydm** }

### 5.931.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.932 std::time\_get<\_CharT, \_InIter> Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::time\_get<\_CharT, \_InIter>:



### Public Types

- enum `dateorder` {  
  `no_order`, `dmy`, `mdy`, `ymd`,  
  `ydm`}
- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

### Public Member Functions

- `time_get` (`size_t` \_\_refs=0)
- `dateorder` `date_order` () const
- `iter_type` `get` (`iter_type` \_\_s, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \*\_\_tm, `char` \_\_format, `char` \_\_modifier=0) const
- `iter_type` `get` (`iter_type` \_\_s, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \*\_\_tm, const `char_type` \*\_\_fmt, const `char_type` \*\_\_fmtend) const
- `iter_type` `get_date` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, `tm` \*\_\_tm) const



- `iter_type get_monthname (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- `iter_type get_time (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- `iter_type get_weekday (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- `iter_type get_year (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`

### Static Public Attributes

- static `locale::id id`

### Protected Member Functions

- virtual `~time_get ()`
- `iter_type _M_extract_name (iter_type __beg, iter_type __end, int & __member, const _CharT ** __names, size_t __indexlen, ios_base & __io, ios_base::iostate & __err) const`
- `iter_type _M_extract_num (iter_type __beg, iter_type __end, int & __member, int __min, int __max, size_t __len, ios_base & __io, ios_base::iostate & __err) const`
- `iter_type _M_extract_via_format (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm, const _CharT * __format) const`
- `iter_type _M_extract_via_format (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm, const _CharT * __format, __time_get_state & __state) const`
- `iter_type _M_extract_wday_or_month (iter_type __beg, iter_type __end, int & __member, const _CharT ** __names, size_t __indexlen, ios_base & __io, ios_base::iostate & __err) const`
- virtual dateorder `do_date_order () const`
- `iter_type do_get (iter_type __s, iter_type __end, ios_base & __f, ios_base::iostate & __err, tm * __tm, char __format, char __modifier) const`
- virtual `iter_type do_get_date (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- virtual `iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- virtual `iter_type do_get_time (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- virtual `iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- virtual `iter_type do_get_year (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`

### Static Protected Member Functions

- static `_c_locale _S_clone_c_locale (_c_locale & __cloc) throw ()`
- static void `_S_create_c_locale (_c_locale & __cloc, const char * __s, _c_locale __old=0)`
- static void `_S_destroy_c_locale (_c_locale & __cloc)`
- static `_c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `_c_locale _S_lc_ctype_c_locale (_c_locale __cloc, const char * __s)`

### 5.932.1 Detailed Description

`template<typename _CharT, typename _InIter>`  
`class std::time_get< _CharT, _InIter >`

Primary class template `time_get`.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

The time\_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time\_get facet.

### 5.932.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT, typename _InIter>
typedef _CharT std::time_get< _CharT, _InIter >::char_type
Public typedefs.
```

#### iter\_type

```
template<typename _CharT, typename _InIter>
typedef _InIter std::time_get< _CharT, _InIter >::iter_type
Public typedefs.
```

### 5.932.3 Constructor & Destructor Documentation

#### time\_get()

```
template<typename _CharT, typename _InIter>
std::time_get< _CharT, _InIter >::time_get (
    size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

#### ~time\_get()

```
template<typename _CharT, typename _InIter>
virtual std::time_get< _CharT, _InIter >::~~time_get () [inline], [protected], [virtual]
Destructor.
```

### 5.932.4 Member Function Documentation

#### date\_order()

```
template<typename _CharT, typename _InIter>
dateorder std::time_get< _CharT, _InIter >::date_order () const [inline]
```

Return preferred order of month, day, and year.

This function returns an enum from time\_base::dateorder giving the preferred ordering if the format x given to time\_get::put::put() only uses month, day, and year. If the format x for the associated locale uses other fields, this function returns time\_base::dateorder::noorder.

NOTE: The library always returns noorder at the moment.

#### Returns

A member of time\_base::dateorder.

References [do\\_date\\_order\(\)](#).

**do\_date\_order()**

```
template<typename _CharT, typename _InIter>
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_↵
date_order () const [protected], [virtual]
```

Return preferred order of month, day, and year.

This function returns an enum from `time_base::dateorder` giving the preferred ordering if the format `x` given to `time_↵`  
`put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

**Returns**

A member of `time_base::dateorder`.

Referenced by [date\\_order\(\)](#).

**do\_get()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get (
    iter_type __s,
    iter_type __end,
    ios_base & __f,
    ios_base::iostate & __err,
    tm * __tm,
    char __format,
    char __modifier) const [inline], [protected]
```

Parse input string according to format.

This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__s</code>        | Start of string to parse.        |
| <code>__end</code>      | End of string to parse.          |
| <code>__f</code>        | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

**Returns**

Iterator to first char not parsed.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base<](#)

Referenced by [get\(\)](#), and [get\(\)](#).

**do\_get\_date()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_date (
```

```

    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [protected], [virtual]

```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get\\_date\(\)](#) for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond date string.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_date\(\)](#).

#### do\_get\_monthname()

```

template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_monthname (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [protected], [virtual]

```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get\\_monthname\(\)](#) for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond month name.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_monthname\(\)](#).

**do\_get\_time()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_time (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [protected], [virtual]
```

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

**See also**

[get\\_time\(\)](#) for details.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |

**Returns**

Iterator to first char beyond time string.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_time\(\)](#).

**do\_get\_weekday()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [protected], [virtual]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

**See also**

[get\\_weekday\(\)](#) for details.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond weekday name.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_weekday\(\)](#).

**do\_get\_year()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_year (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [protected], [virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

## See also

[get\\_year\(\)](#) for details.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond year.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::\\_\\_ctype\\_abstract\\_base< \\_](#) and [std::use\\_facet\(\)](#).

Referenced by [get\\_year\(\)](#).

**get()** [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get (
```

```

iter_type __s,
iter_type __end,
ios_base & __io,
ios_base::iostate & __err,
tm * __tm,
char __format,
char __modifier = 0) const [inline]

```

Parse input string according to format.

This function calls `time_get::do_get` with the provided parameters.

See also

`do_get()` and `get()`.

#### Parameters

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__s</code>        | Start of string to parse.        |
| <code>__end</code>      | End of string to parse.          |
| <code>__io</code>       | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

#### Returns

Iterator to first char not parsed.

References [do\\_get\(\)](#).

#### get() [2/2]

```

template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::get (
    iter_type __s,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm,
    const char_type * __fmt,
    const char_type * __fmtend) const [inline]

```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of `time_put::put`. The format string follows the format specified for `strtime(3)/strptime(3)`. The actual parsing is done by `time_get::do_get`.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__s</code>   | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |
| <code>__fmt</code> | Start of the format string.      |

|                       |                           |
|-----------------------|---------------------------|
| <code>__fmtend</code> | End of the format string. |
|-----------------------|---------------------------|

### Returns

Iterator to first char not parsed.

References [std::ios\\_base::M\\_getloc\(\)](#), [do\\_get\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::is\(\)](#), [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::narrow\(\)](#), [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::towlower\(\)](#), [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::toupper\(\)](#), and [std::use\\_facet\(\)](#).

### get\_date()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get<_CharT, _InIter>::get_date (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [inline]
```

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |

### Returns

Iterator to first char beyond date string.

References [do\\_get\\_date\(\)](#).

### get\_monthname()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get<_CharT, _InIter>::get_monthname (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [inline]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_monthname()`.



Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

#### Parameters

|                    |                                               |
|--------------------|-----------------------------------------------|
| <code>__beg</code> | Start of string to parse.                     |
| <code>__end</code> | End of string to parse.                       |
| <code>__io</code>  | Source of the locale.                         |
| <code>__err</code> | Error flags to set.                           |
| <code>__tm</code>  | Pointer to struct <code>tm</code> to fill in. |

#### Returns

Iterator to first char beyond month name.

References [do\\_get\\_monthname\(\)](#).

### get\_time()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_time (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [inline]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct `tm`. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

#### Parameters

|                    |                                               |
|--------------------|-----------------------------------------------|
| <code>__beg</code> | Start of string to parse.                     |
| <code>__end</code> | End of string to parse.                       |
| <code>__io</code>  | Source of the locale.                         |
| <code>__err</code> | Error flags to set.                           |
| <code>__tm</code>  | Pointer to struct <code>tm</code> to fill in. |

#### Returns

Iterator to first char beyond time string.

References [do\\_get\\_time\(\)](#).

### get\_weekday()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_weekday (
```

```

iter_type __beg,
iter_type __end,
ios_base & __io,
ios_base::iostate & __err,
tm * __tm) const [inline]

```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_weekday().

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond weekday name.

References [do\\_get\\_weekday\(\)](#).

#### get\_year()

```

template<typename _CharT, typename _InIter>
iter_type std::time_get<_CharT, _InIter>::get_year (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [inline]

```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond year.

References [do\\_get\\_year\(\)](#).

**5.932.5 Member Data Documentation****id**

```
template<typename _CharT, typename _InIter>
locale::id std::time_get< _CharT, _InIter >::id [static]
Numpunct facet id.
```

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

**5.933 std::time\_get\_byname< \_CharT, \_InIter > Class Template Reference**

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::time\_get\_byname< \_CharT, \_InIter >:

**Public Types**

- typedef `_CharT` **char\_type**
- enum **dateorder** {  
  **no\_order** , **dmy** , **mdy** , **ymd** ,  
  **ydm** }
- typedef `_InIter` **iter\_type**

**Public Member Functions**

- **time\_get\_byname** (const char \*, size\_t \_\_refs=0)

- **time\_get\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- dateorder [date\\_order](#) () const
- iter\_type [get](#) (iter\_type \_\_s, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, char \_\_format, char \_\_modifier=0) const
- iter\_type [get](#) (iter\_type \_\_s, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, const char\_type \*\_\_fmt, const char\_type \*\_\_fmtend) const
- iter\_type [get\\_date](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- iter\_type [get\\_monthname](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- iter\_type [get\\_time](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- iter\_type [get\\_weekday](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- iter\_type [get\\_year](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- iter\_type **M\_extract\_name** (iter\_type \_\_beg, iter\_type \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- iter\_type **M\_extract\_num** (iter\_type \_\_beg, iter\_type \_\_end, int &\_\_member, int \_\_min, int \_\_max, size\_t \_\_len, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- iter\_type **M\_extract\_via\_format** (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, const \_CharT \*\_\_format) const
- iter\_type **M\_extract\_via\_format** (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, const \_CharT \*\_\_format, \_\_time\_get\_state &\_\_state) const
- iter\_type **M\_extract\_wday\_or\_month** (iter\_type \_\_beg, iter\_type \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- virtual dateorder [do\\_date\\_order](#) () const
- iter\_type [do\\_get](#) (iter\_type \_\_s, iter\_type \_\_end, [ios\\_base](#) &\_\_f, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, char \_\_format, char \_\_modifier) const
- virtual iter\_type [do\\_get\\_date](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual iter\_type [do\\_get\\_monthname](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual iter\_type [do\\_get\\_time](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual iter\_type [do\\_get\\_weekday](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual iter\_type [do\\_get\\_year](#) (iter\_type \_\_beg, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

### Static Protected Member Functions

- static \_\_c\_locale **S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **S\_get\_c\_locale** ()
- static const char \* **S\_get\_c\_name** () throw ()
- static \_\_c\_locale **S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### 5.933.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::time_get_byname< _CharT, _InIter >
```

class time\_get\_byname [22.2.5.2].

### 5.933.2 Member Function Documentation

#### date\_order()

```
template<typename _CharT, typename _InIter>
dateorder std::time_get< _CharT, _InIter >::date_order () const [inline], [inherited]
```

Return preferred order of month, day, and year.

This function returns an enum from time\_base::dateorder giving the preferred ordering if the format x given to time\_↵put::put() only uses month, day, and year. If the format x for the associated locale uses other fields, this function returns time\_base::dateorder::noorder.

NOTE: The library always returns noorder at the moment.

#### Returns

A member of time\_base::dateorder.

References [do\\_date\\_order\(\)](#).

#### do\_date\_order()

```
template<typename _CharT, typename _InIter>
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_↵
date_order () const [protected], [virtual], [inherited]
```

Return preferred order of month, day, and year.

This function returns an enum from time\_base::dateorder giving the preferred ordering if the format x given to time\_↵put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

#### Returns

A member of time\_base::dateorder.

Referenced by [date\\_order\(\)](#).

#### do\_get()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get (
    iter_type __s,
    iter_type __end,
    ios_base & __f,
    ios_base::iostate & __err,
    tm * __tm,
    char __format,
    char __modifier) const [inline], [protected], [inherited]
```

Parse input string according to format.

This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

## Parameters

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__s</code>        | Start of string to parse.        |
| <code>__end</code>      | End of string to parse.          |
| <code>__f</code>        | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

## Returns

Iterator to first char not parsed.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base<\\_CharT, \\_InIter>](#).  
Referenced by [get\(\)](#), and [get\(\)](#).

**do\_get\_date()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_date (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [protected], [virtual], [inherited]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

## See also

[get\\_date\(\)](#) for details.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond date string.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), and [std::use\\_facet\(\)](#).  
Referenced by [get\\_date\(\)](#).

**do\_get\_monthname()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_monthname (
```

```

iter_type __beg,
iter_type __end,
ios_base & __io,
ios_base::iostate & __err,
tm * __tm) const [protected], [virtual], [inherited]

```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

`get_monthname()` for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond month name.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_monthname\(\)](#).

#### do\_get\_time()

```

template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_time (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [protected], [virtual], [inherited]

```

Parse input time string.

This function parses a time according to the format `x` and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

`get_time()` for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond time string.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_time\(\)](#).

**do\_get\_weekday()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_weekday (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [protected], [virtual], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

[get\\_weekday\(\)](#) for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond weekday name.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_weekday\(\)](#).

**do\_get\_year()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_year (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [protected], [virtual], [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

[get\\_year\(\)](#) for details.



**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond year.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::\\_\\_ctype\\_abstract\\_base<\\_](#) and [std::use\\_facet\(\)](#).

Referenced by [get\\_year\(\)](#).

**get()** [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get (
    iter_type __s,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm,
    char __format,
    char __modifier = 0) const [inline], [inherited]
```

Parse input string according to format.

This function calls `time_get::do_get` with the provided parameters.

**See also**

`do_get()` and `get()`.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__s</code>        | Start of string to parse.        |
| <code>__end</code>      | End of string to parse.          |
| <code>__io</code>       | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

**Returns**

Iterator to first char not parsed.

References [do\\_get\(\)](#).

**get()** [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::get (
    iter_type __s,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm,
    const char_type * __fmt,
    const char_type * __fmtend) const [inline], [inherited]
```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of `time_put::put`. The format string follows the format specified for `strptime(3)/strptime(3)`. The actual parsing is done by `time_get::do_get`.

**Parameters**

|                       |                                  |
|-----------------------|----------------------------------|
| <code>__s</code>      | Start of string to parse.        |
| <code>__end</code>    | End of string to parse.          |
| <code>__io</code>     | Source of the locale.            |
| <code>__err</code>    | Error flags to set.              |
| <code>__tm</code>     | Pointer to struct tm to fill in. |
| <code>__fmt</code>    | Start of the format string.      |
| <code>__fmtend</code> | End of the format string.        |

**Returns**

Iterator to first char not parsed.

References `std::ios_base::M_getloc()`, `do_get()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::__ctype_abstract_base<_CharT>::is()`, `std::__ctype_abstract_base<_CharT>::narrow()`, `std::__ctype_abstract_base<_CharT>::toupper()`, and `std::use_facet()`.

**get\_date()**

```
template<typename _CharT, typename _InIter>
iter_type std::time_get<_CharT, _InIter>::get_date (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [inline], [inherited]
```

Parse input date string.

This function parses a date according to the format `x` and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format `x`, `tm` will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

**Parameters**

|                    |                           |
|--------------------|---------------------------|
| <code>__beg</code> | Start of string to parse. |
| <code>__end</code> | End of string to parse.   |

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond date string.

References [do\\_get\\_date\(\)](#).

**get\_monthname()**

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_monthname (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [inline], [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond month name.

References [do\\_get\\_monthname\(\)](#).

**get\_time()**

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_time (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [inline], [inherited]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |

#### Returns

Iterator to first char beyond time string.

References [do\\_get\\_time\(\)](#).

### get\_weekday()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_weekday (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [inline], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |

#### Returns

Iterator to first char beyond weekday name.

References [do\\_get\\_weekday\(\)](#).

**get\_year()**

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_year (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm) const [inline], [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond year.

References [do\\_get\\_year\(\)](#).

**5.933.3 Member Data Documentation****id**

```
template<typename _CharT, typename _InIter>
locale::id std::time_get< _CharT, _InIter >::id [static], [inherited]
Numpunct facet id.
```

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

**5.934 std::chrono::time\_point< \_Clock, \_Dur > Class Template Reference****Public Types**

- typedef `_Clock` **clock**
- typedef `_Dur` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**

**Public Member Functions**

- constexpr **time\_point** (const duration &\_\_dur)
- template<typename \_Dur2, typename = \_Require<is\_convertible<\_Dur2, \_Dur>>>>
 constexpr **time\_point** (const [time\\_point](#)< clock, \_Dur2 > &\_\_t)

- constexpr [time\\_point](#) & **operator++** ()
- constexpr [time\\_point](#) **operator++** (int)
- constexpr [time\\_point](#) & **operator+=** (const duration &\_\_dur)
- constexpr [time\\_point](#) & **operator--** ()
- constexpr [time\\_point](#) **operator--** (int)
- constexpr [time\\_point](#) & **operator-=** (const duration &\_\_dur)
- constexpr duration **time\_since\_epoch** () const

### Static Public Member Functions

- static constexpr [time\\_point](#) **max** () noexcept
- static constexpr [time\\_point](#) **min** () noexcept

### Related Symbols

(Note that these are not member symbols.)

- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>  
constexpr time\_point< _Clock, typename common\_type< _Dur1, duration< _Rep2, _Period2 > >::type >  
operator+ (const time\_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`

#### 5.934.1 Detailed Description

**template<typename \_Clock, typename \_Dur>**  
**class std::chrono::time\_point< \_Clock, \_Dur >**

`chrono::time_point` represents a point in time as measured by a clock  
 The documentation for this class was generated from the following file:

- [chrono.h](#)

### 5.935 std::time\_put<\_CharT, \_OutIter> Class Template Reference

`#include <locale_facets_nonio.h>`

Inheritance diagram for `std::time_put<_CharT, _OutIter >`:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)

## Public Member Functions

- [time\\_put](#) (size\_t \_\_refs=0)
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod=0) const
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, const \_CharT \*\_\_beg, const \_CharT \*\_\_end) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~time\\_put](#) ()
- virtual [iter\\_type do\\_put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

## Static Protected Member Functions

- static `_c_locale` [\\_S\\_clone\\_c\\_locale](#) (`_c_locale` &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (`_c_locale` &\_\_cloc, const char \*\_\_s, `_c_locale` \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (`_c_locale` &\_\_cloc)
- static `_c_locale` [\\_S\\_get\\_c\\_locale](#) ()

- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_type\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### 5.935.1 Detailed Description

**template<typename \_CharT, typename \_OutIter>**  
**class std::time\_put<\_CharT, \_OutIter>**

Primary class template time\_put.

This facet encapsulates the code to format and output dates and times according to formats used by strftime().

The time\_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time\_put facet.

### 5.935.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::time_put<_CharT, _OutIter>::char_type
```

Public typedefs.

#### iter\_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::time_put<_CharT, _OutIter>::iter_type
```

Public typedefs.

### 5.935.3 Constructor & Destructor Documentation

#### time\_put()

```
template<typename _CharT, typename _OutIter>
std::time_put<_CharT, _OutIter>::time_put (
    size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

#### ~time\_put()

```
template<typename _CharT, typename _OutIter>
virtual std::time_put<_CharT, _OutIter>::~~time_put () [inline], [protected], [virtual]
```

Destructor.

### 5.935.4 Member Function Documentation

#### do\_put()

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put<_CharT, _OutIter>::do_put (
```



```

iter_type __s,
ios_base & __io,
char_type __fill,
const tm * __tm,
char __format,
char __mod) const [protected], [virtual]

```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

[put\(\)](#) for more details.

#### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | <code>char_type</code> to use for padding.   |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

#### Returns

Iterator after writing.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::widen\(\)](#).

Referenced by [put\(\)](#), and [put\(\)](#).

#### put() [1/2]

```

template<typename _CharT, typename _OutIter>
iter_type std::time_put<_CharT, _OutIter>::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    char __format,
    char __mod = 0) const [inline]

```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by [strftime\(\)](#). It does so by returning [time\\_put::do\\_put\(\)](#).

#### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | <code>char_type</code> to use for padding.   |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

**Returns**

Iterator after writing.

References [do\\_put\(\)](#).

**put()** [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    const _CharT * __beg,
    const _CharT * __end) const
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The stream to write to.                      |
| <code>__io</code>   | Source of locale.                            |
| <code>__fill</code> | char_type to use for padding.                |
| <code>__tm</code>   | Struct tm with date and time info to format. |
| <code>__beg</code>  | Start of format string.                      |
| <code>__end</code>  | End of format string.                        |

**Returns**

Iterator after writing.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [do\\_put\(\)](#), [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::narrow\(\)](#), and [std::use\\_facet\(\)](#).

**5.935.5 Member Data Documentation****id**

```
template<typename _CharT, typename _OutIter>
locale::id std::time_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

**5.936 std::time\_put\_byname< \_CharT, \_OutIter > Class Template Reference**

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for `std::time_put_byname<_CharT, _OutIter>`:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `_OutIter` **iter\_type**

### Public Member Functions

- **time\_put\_byname** (const char \*, size\_t \_\_refs=0)
- **time\_put\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- iter\_type **put** (iter\_type \_\_s, [ios\\_base](#) &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod=0) const
- iter\_type **put** (iter\_type \_\_s, [ios\\_base](#) &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, const \_CharT \*\_\_beg, const \_CharT \*\_\_end) const

### Static Public Attributes

- static [locale::id](#) **id**

### Protected Member Functions

- virtual iter\_type **do\_put** (iter\_type \_\_s, [ios\\_base](#) &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### 5.936.1 Detailed Description

```
template<typename _CharT, typename _OutIter>
class std::time_put_byname<_CharT, _OutIter>
```

class time\_put\_byname [22.2.5.4].

### 5.936.2 Member Function Documentation

#### do\_put()

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put<_CharT, _OutIter>::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    char __format,
    char __mod) const [protected], [virtual], [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

put() for more details.

#### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | char_type to use for padding.                |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

#### Returns

Iterator after writing.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::widen\(\)](#).

Referenced by [put\(\)](#), and [put\(\)](#).

#### put() [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::time_put<_CharT, _OutIter>::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    char __format,
    char __mod = 0) const [inline], [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time\_put::do\_put().

**Parameters**

|                       |                                                           |
|-----------------------|-----------------------------------------------------------|
| <code>__s</code>      | The stream to write to.                                   |
| <code>__io</code>     | Source of locale.                                         |
| <code>__fill</code>   | <code>char_type</code> to use for padding.                |
| <code>__tm</code>     | Struct <code>tm</code> with date and time info to format. |
| <code>__format</code> | Format char.                                              |
| <code>__mod</code>    | Optional modifier char.                                   |

**Returns**

Iterator after writing.

References [do\\_put\(\)](#).

**put()** [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    const _CharT * __beg,
    const _CharT * __end) const [inherited]
```

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format string. The format string is interpreted as by `strftime()`.

**Parameters**

|                     |                                                           |
|---------------------|-----------------------------------------------------------|
| <code>__s</code>    | The stream to write to.                                   |
| <code>__io</code>   | Source of locale.                                         |
| <code>__fill</code> | <code>char_type</code> to use for padding.                |
| <code>__tm</code>   | Struct <code>tm</code> with date and time info to format. |
| <code>__beg</code>  | Start of format string.                                   |
| <code>__end</code>  | End of format string.                                     |

**Returns**

Iterator after writing.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [do\\_put\(\)](#), [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::narrow\(\)](#), and [std::use\\_facet\(\)](#).

**5.936.3 Member Data Documentation****id**

```
template<typename _CharT, typename _OutIter>
locale::id std::time_put< _CharT, _OutIter >::id [static], [inherited]
```

Numpunct facet id.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.937 `std::timed_mutex` Class Reference

```
#include <mutex>
```

### Public Types

- typedef `__native_type *` **native\_handle\_type**

### Public Member Functions

- **timed\_mutex** (const [timed\\_mutex](#) &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** () noexcept
- [timed\\_mutex](#) & **operator=** (const [timed\\_mutex](#) &)=delete
- bool **try\_lock** () noexcept
- template<class `_Rep`, class `_Period`>  
bool **try\_lock\_for** (const [chrono::duration](#)< `_Rep`, `_Period` > &\_\_rtime)
- template<class `_Clock`, class `_Duration`>  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_atime)
- void **unlock** ()

### Friends

- class `__timed_mutex_impl`< [timed\\_mutex](#) >

#### 5.937.1 Detailed Description

The standard timed mutex type.

A non-recursive mutex that supports a timeout when trying to acquire the lock.

Since

C++11

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.938 `std::to_chars_result` Struct Reference

```
#include <charconv>
```

### Public Attributes

- `errc` **ec**
- `char *` **ptr**

### Friends

- bool **operator==** (const [to\\_chars\\_result](#) &, const [to\\_chars\\_result](#) &)=default

#### 5.938.1 Detailed Description

Result type of `std::to_chars`.

The documentation for this struct was generated from the following file:

- [charconv](#)

### 5.939 `std::chrono::treat_as_floating_point<_Rep>` Struct Template Reference

```
#include <chrono.h>
```

Inheritance diagram for `std::chrono::treat_as_floating_point<_Rep>`:



#### 5.939.1 Detailed Description

```
template<typename _Rep>
struct std::chrono::treat_as_floating_point<_Rep>
```

Trait indicating whether to treat a type as a floating-point type.

The chrono library uses this trait to tell whether a `duration` can represent fractional values of the given precision, or only integral values.

You should specialize this trait for your own numeric types that are used with `duration` and can represent non-integral values.

Since

C++11

The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 5.940 `__gnu_pbds::tree<Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc>` Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`:



## Public Types

- typedef Cmp\_Fn `cmp_fn`
- typedef Node\_Update `node_update`

## Public Member Functions

- `tree` (const `cmp_fn` &c)
- `tree` (const `tree` &other)
- template<typename It>  
`tree` (It first, It last)
- template<typename It>  
`tree` (It first, It last, const `cmp_fn` &c)
- `tree` & `operator=` (const `tree` &other)
- void `swap` (`tree` &other)

### 5.940.1 Detailed Description

```

template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_
tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class
Node_Update = null_node_update, typename _Alloc = std::allocator<char>>>
class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >

```

A tree-based container.



## Template Parameters

|                    |                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                                                                                             |
| <i>Mapped</i>      | Map type.                                                                                                                             |
| <i>Cmp_Fn</i>      | Comparison functor.                                                                                                                   |
| <i>Tag</i>         | Instantiating data structure type, see <code>container_tag</code> .                                                                   |
| <i>Node_Update</i> | Updates tree internal-nodes, restores invariants when invalidated. XXX See <code>design::tree-based-containersnode</code> invariants. |
| <i>_Alloc</i>      | Allocator type.                                                                                                                       |

Base tag choices are: `ov_tree_tag`, `rb_tree_tag`, `splay_tree_tag`.

Base is `basic_branch`.

## 5.940.2 Member Typedef Documentation

**cmp\_fn**

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↵
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
typedef Cmp_Fn \_\_gnu\_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn
Comparison functor type.
```

## 5.940.3 Constructor &amp; Destructor Documentation

**tree()** [1/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↵
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
\_\_gnu\_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
    const cmp\_fn & c) [inline]
```

Constructor taking some policy objects. `r_cmp_fn` will be copied by the `Cmp_Fn` object of the container object.

**tree()** [2/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↵
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It>
\_\_gnu\_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
    It first,
    It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

**tree()** [3/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↵
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It>
\_\_gnu\_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
    It first,
```

```

        It last,
        const cmp_fn & c) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_cmp_fn` will be copied by the `cmp_fn` object of the container object. The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.941 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >` Struct Template Reference

### 5.941.1 Detailed Description

```

template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >

```

Tree metadata helper.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

## 5.942 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >` Struct Template Reference

```

#include <node_metadata_selector.hpp>

```

### Public Types

- typedef `Node_Update::metadata_type` **type**

### 5.942.1 Detailed Description

```

template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >

```

Specialization, false.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

## 5.943 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >` Struct Template Reference

```

#include <node_metadata_selector.hpp>

```

### Public Types

- typedef `null_type` **type**

### 5.943.1 Detailed Description

```

template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >

```

Specialization, true.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

### 5.944 `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >` Struct Template Reference

```
#include <node_metadata_selector.hpp>
```

#### Public Types

- typedef `tree_metadata_helper< __node_u, null_update >::type` **type**

#### 5.944.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename
Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Tree node metadata dispatch.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

### 5.945 `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >` Class Template Reference

```
#include <tree_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`:



#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**

- typedef size\_type **metadata\_type**
- typedef Node\_CItr **node\_const\_iterator**
- typedef Node\_Itr **node\_iterator**
- typedef allocator\_type::size\_type **size\_type**

### Public Member Functions

- iterator [find\\_by\\_order](#) (size\_type)
- const\_iterator [find\\_by\\_order](#) (size\_type) const
- size\_type [order\\_of\\_key](#) (key\_const\_reference) const

### Protected Member Functions

- void [operator\(\)](#) (node\_iterator, node\_const\_iterator) const

#### 5.945.1 Detailed Description

**template<typename Node\_CItr, typename Node\_Itr, typename Cmp\_Fn, typename \_Alloc>**  
**class `__gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >`**

Functor updating ranks of entrees.

#### 5.945.2 Member Function Documentation

##### [find\\_by\\_order\(\)](#) [1/2]

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree\_order\_statistics\_node\_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::iterator \_\_gnu\_pbds::tree\_order\_statist
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
    size_type order) [inline]
```

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

##### [find\\_by\\_order\(\)](#) [2/2]

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree\_order\_statistics\_node\_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::const_iterator \_\_gnu\_pbds::tree\_order\_s
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
    size_type order) const [inline]
```

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

##### [operator\(\)\(\)](#)

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
void \_\_gnu\_pbds::tree\_order\_statistics\_node\_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::operator()
(
    node_iterator node_it,
    node_const_iterator end_nd_it) const [inline], [protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

##### [order\\_of\\_key\(\)](#)

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree\_order\_statistics\_node\_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::size_type \_\_gnu\_pbds::tree\_order\_statist
```

```
Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::order_of_key (
    key_const_reference r_key) const [inline]
```

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

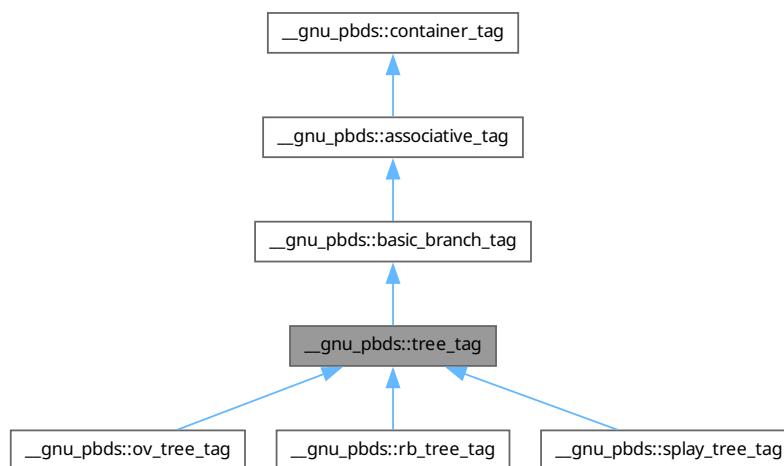
The documentation for this class was generated from the following file:

- [tree\\_policy.hpp](#)

## 5.946 `__gnu_pbds::tree_tag` Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::tree_tag`:



### 5.946.1 Detailed Description

Basic tree structure.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.947 `__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >` Struct Template Reference

### 5.947.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >
```

Tree traits class, primary template.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

## 5.948 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct` Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef `tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type` `metadata_type`
- typedef `ov_tree_node_const_it< value_type, metadata_type, _Alloc >` `node_const_iterator`
- typedef `ov_tree_node_it< value_type, metadata_type, _Alloc >` `node_iterator`
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` `node_update`
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`

### 5.948.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr,
class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Tree traits.

### 5.948.2 Member Typedef Documentation

#### `node_const_iterator`

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr,
class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef ov_tree_node_const_it< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)

## 5.949 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct` Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >:`



## Public Types

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

### 5.949.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
```

Specialization.

### 5.949.2 Member Typedef Documentation

#### **node\_const\_iterator**

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_traits.hpp](#)

### 5.950 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



## Public Types

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

### 5.950.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
```

```
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
```

Specialization.



## 5.950.2 Member Typedef Documentation

### node\_const\_iterator

```
typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >↵
::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >↵
::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↵
Alloc >, _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/traits.hpp](#)

## 5.951 \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc > Struct Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef [tree\\_node\\_metadata\\_dispatch](#)< Key, [null\\_type](#), Cmp\_Fn, Node\_Update, \_Alloc >::type **metadata\_type**
- typedef [ov\\_tree\\_node\\_const\\_it\\_](#)< [value\\_type](#), metadata\_type, \_Alloc > **node\_const\_iterator**
- typedef [node\\_const\\_iterator](#) **node\_iterator**
- typedef Node\_Update< [node\\_const\\_iterator](#), [node\\_const\\_iterator](#), Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#)< [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > \* **null\_node\_↵  
\_update\_pointer**

### 5.951.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_↵
, typename _Alloc_ > class Node_Update, typename _Alloc>
```

```
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Specialization.

## 5.951.2 Member Typedef Documentation

### node\_const\_iterator

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_↵
_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef ov_tree_node_const_it_< value\_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, null\_type, Cmp_Fn, Node_Update, ov\_tree\_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)

## 5.952 \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



## Public Types

- typedef `bin_search_tree_const_it` < typename `node_alloc_traits::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, false, `_Alloc` > **const\_reverse\_iterator**
- typedef `rb_tree_node` < `types_traits< Key, null_type, _Alloc, false >::value_type`, `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type`, `_Alloc` > **node**
- typedef `bin_search_tree_const_node_it` < `rb_tree_node` < `types_traits< Key, null_type, _Alloc, false >::value_type`, `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type`, `_Alloc` >, `point_const_iterator`, `point_iterator`, `_Alloc` > **node\_const\_iterator**
- typedef `bin_search_tree_node_it` < `rb_tree_node` < `types_traits< Key, null_type, _Alloc, false >::value_type`, `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type`, `_Alloc` >, `point_const_iterator`, `point_iterator`, `_Alloc` > **node\_iterator**
- typedef `Node_Update` < `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > **node\_update**
- typedef `__gnu_pbds::null_node_update` < `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > \* **null\_node\_update\_pointer**
- typedef `bin_search_tree_const_it` < typename `node_alloc_traits::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, true, `_Alloc` > **point\_const\_iterator**
- typedef `bin_search_tree_it` < typename `node_alloc_traits::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, true, `_Alloc` > **point\_iterator**
- typedef `bin_search_tree_it` < typename `node_alloc_traits::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, false, `_Alloc` > **reverse\_iterator**

### 5.952.1 Detailed Description

```

template<typename Key, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

```

Specialization.

## 5.952.2 Member Typedef Documentation

### node\_const\_iterator

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >↔
::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, null_type, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, null_type, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type,
_Alloc >, _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/traits.hpp](#)

## 5.953 \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >:



## Public Types

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >↔::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value↔_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node↔_update_pointer`

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

### 5.953.1 Detailed Description

`template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_↵  
, typename _Alloc_ > class Node_Update, typename _Alloc>  
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`

Specialization.

### 5.953.2 Member Typedef Documentation

#### `node_const_iterator`

```
typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc,  
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >↵  
::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<  
Key, null_type, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, null_type, _Alloc,  
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >↵  
::type, _Alloc >, _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

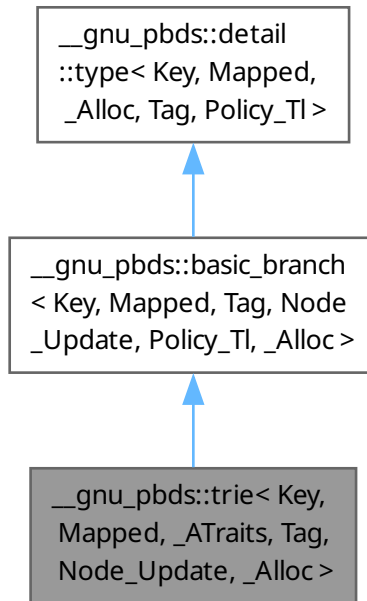
The documentation for this struct was generated from the following file:

- `splay_tree_/traits.hpp`

## 5.954 `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >` Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`:



## Public Types

- typedef `_ATraits` [access\\_traits](#)
- typedef `Node_Update` **node\_update**

## Public Member Functions

- [trie](#) (const [access\\_traits](#) &t)
- **trie** (const [trie](#) &other)
- template<typename It>  
[trie](#) (It first, It last)
- template<typename It>  
[trie](#) (It first, It last, const [access\\_traits](#) &t)
- [trie](#) & **operator=** (const [trie](#) &other)
- void **swap** ([trie](#) &other)

### 5.954.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>>
```

```
class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >
```

A trie-based container.

## Template Parameters

|                    |                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                                                                                             |
| <i>Mapped</i>      | Map type.                                                                                                                             |
| <i>_ATraits</i>    | Element access traits.                                                                                                                |
| <i>Tag</i>         | Instantiating data structure type, see <code>container_tag</code> .                                                                   |
| <i>Node_Update</i> | Updates trie internal-nodes, restores invariants when invalidated. XXX See <code>design::tree-based-containersnode</code> invariants. |
| <i>_Alloc</i>      | Allocator type.                                                                                                                       |

Base tag choice is `pat_trie_tag`.

Base is `basic_branch`.

## 5.954.2 Member Typedef Documentation

**access\_traits**

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type,
        typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_CItr,
        typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
        = std::allocator<char>>
        typedef _ATraits __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_traits
```

Element access traits type.

## 5.954.3 Constructor &amp; Destructor Documentation

**trie()** [1/3]

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type,
        typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_CItr,
        typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
        = std::allocator<char>>
        __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
            const access_traits & t) [inline]
```

Constructor taking some policy objects. `r_access_traits` will be copied by the `_ATraits` object of the container object.

**trie()** [2/3]

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type,
        typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_CItr,
        typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
        = std::allocator<char>>
        template<typename It>
        __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
            It first,
            It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

**trie()** [3/3]

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type,
        typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_CItr,
        typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
```

```
= std::allocator<char>>
template<typename It>
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
    It first,
    It last,
    const access_traits & t) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects. The value\_types between first\_it and last\_it will be inserted into the container object.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

### 5.955 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, \_BTp > Struct Template Reference

#### 5.955.1 Detailed Description

```
template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >
```

Trie metadata helper.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

### 5.956 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, false > Struct Template Reference

```
#include <node_metadata_selector.hpp>
```

#### Public Types

- typedef Node\_Update::metadata\_type type

#### 5.956.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
```

Specialization, false.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

### 5.957 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, true > Struct Template Reference

```
#include <node_metadata_selector.hpp>
```

#### Public Types

- typedef [null\\_type](#) type

#### 5.957.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
```

Specialization, true.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 5.958 `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >` Struct Template Reference

```
#include <node_metadata_selector.hpp>
```

#### Public Types

- typedef [trie\\_metadata\\_helper](#)< `__node_u`, `null_update` >::type `type`

#### 5.958.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename
Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Trie node metadata dispatch.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 5.959 `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

```
#include <trie_policy.hpp>
```



Inheritance diagram for `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



## Public Types

- typedef `access_traits::const_iterator` **a\_const\_iterator**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `size_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

## Public Member Functions

- iterator [find\\_by\\_order](#) (size\_type)
- const\_iterator [find\\_by\\_order](#) (size\_type) const
- size\_type [order\\_of\\_key](#) (key\_const\_reference) const
- size\_type [order\\_of\\_prefix](#) (a\_const\_iterator, a\_const\_iterator) const

## Protected Member Functions

- void `operator()` (node\_iterator, node\_const\_iterator) const

## Private Member Functions

- virtual const\_iterator `end` () const =0
- virtual const access\_traits & `get_access_traits` () const =0

### 5.959.1 Detailed Description

`template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>`  
`class __gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >`

Functor updating ranks of entrees.

### 5.959.2 Member Function Documentation

#### `find_by_order()` [1/2]

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::iterator
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::find_by_order (
    size_type order) [inline]
```

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

#### `find_by_order()` [2/2]

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::const_iterator
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::find_by_order (
    size_type order) const [inline]
```

Finds an entry by `__order`. Returns a const\_iterator to the entry with the `__order` order, or a const\_iterator to the container object's end if order is at least the size of the container object.

#### `operator>()`

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::
operator() (
    node_iterator nd_it,
    node_const_iterator ) const [inline], [protected]
```

Updates the rank of a node through a node\_iterator node\_it; end\_nd\_it is the end node iterator.

#### `order_of_key()`

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::size_type
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::order_of_key (
    key_const_reference r_key) const [inline]
```

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

### order\_of\_prefix()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type __gnu_pbds::trie_order_statistics_node_update<
Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (
    a_const_iterator b,
    a_const_iterator e) const [inline]
```

Returns the order of a prefix within a sequence. For example, if [b, e) is the smallest prefix, this method will return 0; if r\_key is a key between the smallest and next key, this method will return 1; if r\_key is a key larger than the largest key, this method will return the size of r\_c.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

## 5.960 \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_CItr, Node\_Itr, \_ATraits, \_Alloc > Class Template Reference

```
#include <trie_policy_base.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_CItr, Node\_Itr, \_ATraits, \_Alloc >:



### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `null_type` **metadata\_type**
- typedef `Node_CItr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

### Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rkey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**

- typedef [rebind\\_traits](#)< \_Alloc, rkey\_type > **rebind\_k**
- typedef [rebind\\_traits](#)< \_Alloc, rvalue\_type > **rebind\_v**
- typedef [rebind\\_v](#)::reference **reference**
- typedef [std::iterator\\_traits](#)< it\_type >::value\_type **value\_type**

### Protected Member Functions

- virtual const\_iterator **end** () const =0
- virtual iterator [end](#) ()=0
- it\_type **end\_iterator** () const
- virtual const access\_traits & **get\_access\_traits** () const =0
- virtual node\_const\_iterator **node\_begin** () const =0
- virtual node\_iterator **node\_begin** ()=0
- virtual node\_const\_iterator **node\_end** () const =0
- virtual node\_iterator **node\_end** ()=0

### Static Protected Member Functions

- static size\_type **common\_prefix\_len** (node\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static key\_const\_reference **extract\_key** (const\_reference r\_val)
- static iterator **leftmost\_it** (node\_iterator)
- static bool **less** (e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static iterator **rightmost\_it** (node\_iterator)

#### 5.960.1 Detailed Description

`template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>`  
**class** `__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >`

Base class for trie policies.

#### 5.960.2 Member Function Documentation

##### **end()**

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
virtual iterator \_\_gnu\_pbds::detail::trie\_policy\_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >↔
::end () [protected], [pure virtual]
```

Implements [\\_\\_gnu\\_pbds::detail::branch\\_policy](#)< Node\_Cltr, Node\_Itr, \_Alloc >.

The documentation for this class was generated from the following file:

- [trie\\_policy\\_base.hpp](#)

#### 5.961 `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

```
#include <trie_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >`:



## Public Types

- typedef `access_traits::const_iterator` [a\\_const\\_iterator](#)
- typedef `_ATraits` [access\\_traits](#)
- typedef `_Alloc` [allocator\\_type](#)
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef [null\\_type](#) **metadata\_type**
- typedef `Node_CItr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` [size\\_type](#)

## Public Member Functions

- `std::pair< iterator, iterator >` [prefix\\_range](#) ([a\\_const\\_iterator](#), [a\\_const\\_iterator](#))
- `std::pair< const_iterator, const_iterator >` [prefix\\_range](#) ([a\\_const\\_iterator](#), [a\\_const\\_iterator](#)) const
- `std::pair< iterator, iterator >` [prefix\\_range](#) (`key_const_reference`)
- `std::pair< const_iterator, const_iterator >` [prefix\\_range](#) (`key_const_reference`) const

## Protected Member Functions

- void `operator()` (`node_iterator node_it`, `node_const_iterator end_nd_it`) const

### 5.961.1 Detailed Description

template<typename `Node_CIttr`, typename `Node_Itr`, typename `_ATraits`, typename `_Alloc`>  
class `__gnu_pbds::trie_prefix_search_node_update`< `Node_CIttr`, `Node_Itr`, `_ATraits`, `_Alloc` >

A node updatator that allows tries to be searched for the range of values that match a certain prefix.

### 5.961.2 Member Typedef Documentation

#### `a_const_iterator`

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef access_traits::const_iterator __gnu_pbds::trie_prefix_search_node_update< Node_CIttr,
Node_Itr, _ATraits, _Alloc >::a_const_iterator
Const element iterator.
```

#### `access_traits`

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _ATraits __gnu_pbds::trie_prefix_search_node_update< Node_CIttr, Node_Itr, _ATraits, _↵
Alloc >::access_traits
Element access traits.
```

#### `allocator_type`

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _Alloc __gnu_pbds::trie_prefix_search_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc
>::allocator_type
_Alloc type.
```

#### `size_type`

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef allocator_type::size_type __gnu_pbds::trie_prefix_search_node_update< Node_CIttr, Node_↵
Itr, _ATraits, _Alloc >::size_type
Size type.
```

### 5.961.3 Member Function Documentation

#### `operator>()`

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::trie_prefix_search_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::operator()
(
    node_iterator node_it,
    node_const_iterator end_nd_it) const [inline], [protected]
```

Called to update a node's metadata.

#### `prefix_range()` [1/4]

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >↵
::iterator, typename trie_prefix_search_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >↵
```

```

::iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
    a_const_iterator b,
    a_const_iterator e)

```

Finds the iterator range corresponding to all values whose prefixes match [b, e).

#### prefix\_range() [2/4]

```

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↵
_Alloc >::prefix_range (
    a_const_iterator b,
    a_const_iterator e) const

```

Finds the const iterator range corresponding to all values whose prefixes match [b, e).

#### prefix\_range() [3/4]

```

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
    key_const_reference r_key)

```

Finds the iterator range corresponding to all values whose prefixes match r\_key.

#### prefix\_range() [4/4]

```

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↵
_Alloc >::prefix_range (
    key_const_reference r_key) const

```

Finds the const iterator range corresponding to all values whose prefixes match r\_key.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

## 5.962 \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc > Struct Template Reference

```
#include <trie_policy.hpp>
```

### Public Types

- enum { **reverse** }
- enum { **min\_e\_val** , **max\_e\_val** , **max\_size** }
- typedef detail::\_\_conditional\_type< Reverse, typenameString::const\_reverse\_iterator, typenameString::const\_↵  
iterator >::\_\_type **const\_iterator**
- typedef std::iterator\_traits< const\_iterator >::value\_type **e\_type**
- typedef detail::rebind\_traits< \_Alloc, key\_type >::const\_reference **key\_const\_reference**
- typedef String **key\_type**
- typedef \_Alloc::size\_type **size\_type**

## Static Public Member Functions

- static `const_iterator begin` (key\_const\_reference)
- static `size_type e_pos` (e\_type e)
- static `const_iterator end` (key\_const\_reference)

### 5.962.1 Detailed Description

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_↵  
traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_↵  
traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>>  
struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >
```

Element access traits for string types.

#### Template Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>String</i>    | String type.                                      |
| <i>Min_E_Val</i> | Minimal element value.                            |
| <i>Max_E_Val</i> | Maximum element value.                            |
| <i>Reverse</i>   | Reverse iteration should be used. Default: false. |
| <i>_Alloc</i>    | Allocator type.                                   |

### 5.962.2 Member Typedef Documentation

#### `const_iterator`

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__↵  
numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail↵  
::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc =  
std::allocator<char>>  
typedef detail::__conditional_type<Reverse,typenameString::const_reverse_iterator,typenameString↵  
::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val,  
Reverse, _Alloc >::const_iterator
```

Element const iterator type.

#### `e_type`

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__↵  
numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail↵  
::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc =  
std::allocator<char>>  
typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits<  
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type
```

Element type.

### 5.962.3 Member Function Documentation

#### `begin()`

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max↵  
_E_Val, bool Reverse, typename _Alloc>  
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_strin↵  
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin (  
key_const_reference r_key) [inline], [static]
```



Returns a `const_iterator` to the first element of `key_const_reference` argument.

### **e\_pos()**

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos (
    e_type e) [inline], [static]
```

Maps an element to a position.

### **end()**

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::end (
    key_const_reference r_key) [inline], [static]
```

Returns a `const_iterator` to the after-last element of `key_const_reference` argument.

The documentation for this struct was generated from the following file:

- [trie\\_policy.hpp](#)

## **5.963 \_\_gnu\_pbds::trie\_tag Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::trie_tag`:



#### 5.963.1 Detailed Description

Basic trie structure.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.964 `__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >` Struct Template Reference

#### 5.964.1 Detailed Description

```
template<typename Key, typename Data, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc> struct __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >
```

Trie traits class, primary template.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

## 5.965 `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::CItr< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::CItr< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::Head< synth_access_traits, metadata >` **head**
- typedef `base_type::Inode< synth_access_traits, metadata >` **inode**
- typedef `base_type::Iter< node, leaf, head, inode, true >` **iterator**
- typedef `base_type::Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, Mapped, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_const\_iterator**
- typedef `base_type::Node_iter< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node\_update**
- typedef `null_node_update< node_const_iterator, node_iterator, _ATraits, _Alloc > *` **null\_node\_update\_pointer**
- typedef `base_type::Iter< node, leaf, head, inode, false >` **reverse\_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, false, access_traits >` **synth\_access\_traits**

### 5.965.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

### 5.965.2 Member Typedef Documentation

#### **node\_const\_iterator**

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef base_type::Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits<
```

```
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

#### **node\_update**

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

## synth\_access\_traits

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef \_\_gnu\_pbds::detail::synth\_access\_traits<type\_traits, false, access\_traits> \_\_gnu\_pbds::detail::trie\_traits
Key, Mapped, _ATraits, Node_Update, pat\_trie\_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

## 5.966 `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::CItr< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::CItr< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::Head< synth_access_traits, metadata >` **head**
- typedef `base_type::Inode< synth_access_traits, metadata >` **inode**
- typedef `const_iterator` **iterator**
- typedef `base_type::Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, null_type, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_const\_iterator**
- typedef `node_const_iterator` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node\_update**
- typedef `null_node_update< node_const_iterator, node_const_iterator, _ATraits, _Alloc > * null_node_update`↵  
**\_pointer**
- typedef `const_reverse_iterator` **reverse\_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, true, access_traits >` **synth\_access\_traits**

### 5.966.1 Detailed Description

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >
```

Specialization.

### 5.966.2 Member Typedef Documentation

#### node\_const\_iterator

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef base\_type::Node\_citer<node, leaf, head, inode, const\_iterator, iterator, \_Alloc> \_\_gnu\_pbds::detail::trie\_traits
Key, null\_type, _ATraits, Node_Update, pat\_trie\_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

## node\_update

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
Type for node update.
```

## synth\_access\_traits

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef __gnu_pbds::detail::synth_access_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
Type for synthesized traits.
```

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

## 5.967 \_\_gnu\_pbds::trivial\_iterator\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

### 5.967.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.968 std::try\_to\_lock\_t Struct Reference

```
#include <std_mutex.h>
```

### 5.968.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

## 5.969 std::tuple<\_Elements> Class Template Reference

```
#include <tuple>
```

### Public Member Functions

- `template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor<_Valid, _UElements...> = true>`  
constexpr **tuple** (`_UElements` &&... `__elements`) noexcept(`__nothrow_constructible<_UElements...>()`)
- `template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor<_Valid, _UElements...> = false>`  
constexpr **tuple** (`_UElements` &&... `__elements`) noexcept(`__nothrow_constructible<_UElements...>()`)
- `template<typename _Alloc, _ImplicitDefaultCtor<is_object<_Alloc>::value> = true>`  
constexpr **tuple** (`allocator_arg_t` `__tag`, const `_Alloc` &`__a`)
- `template<typename _Alloc, _ExplicitDefaultCtor<is_object<_Alloc>::value> = false>`  
constexpr **tuple** (`allocator_arg_t` `__tag`, const `_Alloc` &`__a`)

- `template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor<_Valid, _UElements...> = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, _UElements &&... __elements)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor<_Valid, _UElements...> = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, _UElements &&... __elements)`
- `template<typename _Alloc, bool _NotEmpty = (sizeof...( _Elements) >= 1), _ImplicitCtor<_NotEmpty, const _Elements &...> = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, const _Elements &... __elements)`
- `template<typename _Alloc, bool _NotEmpty = (sizeof...( _Elements) >= 1), _ExplicitCtor<_NotEmpty, const _Elements &...> = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, const _Elements &... __elements)`
- `template<typename _Alloc>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, const tuple & __in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_<←`  
`ctor<const tuple<_UElements...>&>(), _ImplicitCtor<_Valid, const _UElements &...> = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, const tuple<_UElements...> & __in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_<←`  
`ctor<const tuple<_UElements...>&>(), _ExplicitCtor<_Valid, const _UElements &...> = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, const tuple<_UElements...> & __in)`
- `template<typename _Alloc>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, tuple && __in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_<←`  
`ctor<tuple<_UElements...>&&(), _ImplicitCtor<_Valid, _UElements...> = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, tuple<_UElements...> && __in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_<←`  
`ctor<tuple<_UElements...>&&(), _ExplicitCtor<_Valid, _UElements...> = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc & __a, tuple<_UElements...> && __in)`
- `template<bool _NotEmpty = (sizeof...( _Elements) >= 1), _ImplicitCtor<_NotEmpty, const _Elements &...> = true>`  
`constexpr tuple (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &...>())`
- `template<bool _NotEmpty = (sizeof...( _Elements) >= 1), _ExplicitCtor<_NotEmpty, const _Elements &...> = false>`  
`constexpr tuple (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &...>())`
- `constexpr tuple (const tuple &)=default`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_ctor<const tuple<_<←`  
`UElements...>&>(), _ImplicitCtor<_Valid, const _UElements &...> = true>`  
`constexpr tuple (const tuple<_UElements...> & __in) noexcept(__nothrow_constructible< const _UElements &...>())`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_ctor<const tuple<_<←`  
`UElements...>&>(), _ExplicitCtor<_Valid, const _UElements &...> = false>`  
`constexpr tuple (const tuple<_UElements...> & __in) noexcept(__nothrow_constructible< const _UElements &...>())`
- `constexpr tuple (tuple &&)=default`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_ctor<tuple<_<←`  
`UElements...>&&(), _ImplicitCtor<_Valid, _UElements...> = true>`  
`constexpr tuple (tuple<_UElements...> && __in) noexcept(__nothrow_constructible< _UElements...>())`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_ctor<tuple<_<←`  
`UElements...>&&(), _ExplicitCtor<_Valid, _UElements...> = false>`  
`constexpr tuple (tuple<_UElements...> && __in) noexcept(__nothrow_constructible< _UElements...>())`
- `constexpr tuple & operator= (__conditional_t< __assignable< _Elements...>(), tuple &&, __nonesuch && > __in) noexcept(__nothrow_assignable< _Elements...>())`
- `constexpr tuple & operator= (__conditional_t< __assignable< const _Elements &...>(), const tuple &, const __nonesuch & > __in) noexcept(__nothrow_assignable< const _Elements &...>())`

- `template<typename... _UElements>`  
`constexpr __enable_if_t< __assignable< const _UElements &... >(), tuple & > operator= (const tuple< _UElements... > &__in) noexcept(__nothrow_assignable< const _UElements &... >())`
- `template<typename... _UElements>`  
`constexpr __enable_if_t< __assignable< _UElements... >(), tuple & > operator= (tuple< _UElements... > &&__in) noexcept(__nothrow_assignable< _UElements... >())`
- `constexpr void swap (tuple &__in) noexcept(__and< __is_nothrow_swappable< _Elements >... >::value)`

### 5.969.1 Detailed Description

`template<typename... _Elements>`  
**class** `std::tuple< _Elements >`

Primary class template, tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

## 5.970 std::tuple< \_T1, \_T2 > Class Template Reference

```
#include <tuple>
```

### Public Member Functions

- `template<typename _U1, typename _U2, _ImplicitCtor<!__is_alloc_arg< _U1 >(), _U1, _U2 > = true>`  
`constexpr tuple ( _U1 &&__a1, _U2 &&__a2) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `template<typename _U1, typename _U2, _ExplicitCtor<!__is_alloc_arg< _U1 >(), _U1, _U2 > = false>`  
`constexpr tuple ( _U1 &&__a1, _U2 &&__a2) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `constexpr tuple ( _UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())`
- `constexpr tuple ( _UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a)`
- `template<typename _Alloc, _ImplicitDefaultCtor< is_object< _Alloc >::value, _T1, _T2 > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a)`
- `template<typename _Alloc, _ExplicitDefaultCtor< is_object< _Alloc >::value, _T1, _T2 > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a)`
- `template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, _U1 &&__a1, _U2 &&__a2)`
- `template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, _U1 &&__a1, _U2 &&__a2)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const _Elements &&... __elements)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const _Elements &&... __elements)`
- `template<typename _Alloc, bool _Dummy = true, _ImplicitCtor< _Dummy, const _T1 &, const _T2 & > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const _T1 &__a1, const _T2 &__a2)`
- `template<typename _Alloc, bool _Dummy = true, _ExplicitCtor< _Dummy, const _T1 &, const _T2 & > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const _T1 &__a1, const _T2 &__a2)`
- `template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const pair< _U1, _U2 > &__in)`
- `template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const pair< _U1, _U2 > &__in)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)`

- `template<typename _Alloc>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)`
- `template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _U1, _U2 > &__in)`
- `template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _U1, _U2 > &__in)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)`
- `template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)`
- `template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)`
- `template<typename _Alloc>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)`
- `template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)`
- `template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)`
- `constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)`
- `constexpr tuple (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())`
- `constexpr tuple (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())`
- `template<bool _Dummy = true, _ImplicitCtor< _Dummy, const _T1 &, const _T2 & > = true>`  
`constexpr tuple (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())`
- `template<bool _Dummy = true, _ExplicitCtor< _Dummy, const _T1 &, const _T2 & > = false>`  
`constexpr tuple (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>`  
`constexpr tuple (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>`  
`constexpr tuple (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `constexpr tuple (const tuple &)=default`
- `constexpr tuple (const tuple &)=default`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>`  
`constexpr tuple (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>`  
`constexpr tuple (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `constexpr tuple (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())`
- `constexpr tuple (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (pair< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>`  
`constexpr tuple (pair< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())`



- constexpr **tuple** (**tuple** &&)=default
- constexpr **tuple** (**tuple** &&)=default
- template<typename \_U1, typename \_U2, \_ImplicitCtor< true, \_U1, \_U2 > = true>  
constexpr **tuple** (**tuple**< \_U1, \_U2 > &&\_\_in) noexcept(\_\_nothrow\_constructible< \_U1, \_U2 >())
- template<typename \_U1, typename \_U2, \_ExplicitCtor< true, \_U1, \_U2 > = false>  
constexpr **tuple** (**tuple**< \_U1, \_U2 > &&\_\_in) noexcept(\_\_nothrow\_constructible< \_U1, \_U2 >())
- constexpr **tuple** (**tuple**< \_UElements... > &&\_\_in) noexcept(\_\_nothrow\_constructible< \_UElements... >())
- constexpr **tuple** (**tuple**< \_UElements... > &&\_\_in) noexcept(\_\_nothrow\_constructible< \_UElements... >())
- constexpr **tuple** & **operator=** ( \_\_conditional\_t< \_\_assignable< \_Elements... >(), **tuple** &&, \_\_nonesuch && > \_\_in) noexcept(\_\_nothrow\_assignable< \_Elements... >())
- constexpr **tuple** & **operator=** ( \_\_conditional\_t< \_\_assignable< \_T1, \_T2 >(), **tuple** &&, \_\_nonesuch && > \_\_in) noexcept(\_\_nothrow\_assignable< \_T1, \_T2 >())
- constexpr **tuple** & **operator=** ( \_\_conditional\_t< \_\_assignable< const \_Elements &... >(), const **tuple** &, const \_\_nonesuch & > \_\_in) noexcept(\_\_nothrow\_assignable< const \_Elements &... >())
- constexpr **tuple** & **operator=** ( \_\_conditional\_t< \_\_assignable< const \_T1 &, const \_T2 & >(), const **tuple** &, const \_\_nonesuch & > \_\_in) noexcept(\_\_nothrow\_assignable< const \_T1 &, const \_T2 & >())
- template<typename \_U1, typename \_U2>  
constexpr \_\_enable\_if\_t< \_\_assignable< const \_U1 &, const \_U2 & >(), **tuple** & > **operator=** (const **pair**< \_U1, \_U2 > &&\_\_in) noexcept(\_\_nothrow\_assignable< const \_U1 &, const \_U2 & >())
- template<typename \_U1, typename \_U2>  
constexpr \_\_enable\_if\_t< \_\_assignable< const \_U1 &, const \_U2 & >(), **tuple** & > **operator=** (const **tuple**< \_U1, \_U2 > &&\_\_in) noexcept(\_\_nothrow\_assignable< const \_U1 &, const \_U2 & >())
- constexpr \_\_enable\_if\_t< \_\_assignable< const \_UElements &... >(), **tuple** & > **operator=** (const **tuple**< \_↵ \_UElements... > &&\_\_in) noexcept(\_\_nothrow\_assignable< const \_UElements &... >())
- template<typename \_U1, typename \_U2>  
constexpr \_\_enable\_if\_t< \_\_assignable< \_U1, \_U2 >(), **tuple** & > **operator=** (**pair**< \_U1, \_U2 > &&\_\_in) noexcept(\_\_nothrow\_assignable< \_U1, \_U2 >())
- template<typename \_U1, typename \_U2>  
constexpr \_\_enable\_if\_t< \_\_assignable< \_U1, \_U2 >(), **tuple** & > **operator=** (**tuple**< \_U1, \_U2 > &&\_\_in) noexcept(\_\_nothrow\_assignable< \_U1, \_U2 >())
- constexpr \_\_enable\_if\_t< \_\_assignable< \_UElements... >(), **tuple** & > **operator=** (**tuple**< \_UElements... > &&\_\_in) noexcept(\_\_nothrow\_assignable< \_UElements... >())
- constexpr void **swap** (**tuple** &\_\_in) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable< \_Elements >... >::value)
- constexpr void **swap** (**tuple** &\_\_in) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable< \_T1 >, \_\_is\_nothrow\_↵ swappable< \_T2 > >::value)

### 5.970.1 Detailed Description

**template<typename \_T1, typename \_T2>**  
**class std::tuple< \_T1, \_T2 >**

Partial specialization, 2-element tuple. Includes construction and assignment from a pair.  
The documentation for this class was generated from the following file:

- [tuple](#)

## 5.971 std::tuple\_element< \_\_i, \_Tp > Struct Template Reference

### 5.971.1 Detailed Description

**template<size\_t \_\_i, typename \_Tp>**  
**struct std::tuple\_element< \_\_i, \_Tp >**

Gives the type of the ith element of a given tuple type.

The documentation for this struct was generated from the following file:

- [utility.h](#)

## 5.972 `std::tuple_element< 0, pair< _Tp1, _Tp2 > >` Struct Template Reference

```
#include <stl_pair.h>
```

### Public Types

- `typedef _Tp1 type`

#### 5.972.1 Detailed Description

```
template<class _Tp1, class _Tp2>  
struct std::tuple_element< 0, pair< _Tp1, _Tp2 > >
```

Partial specialization for `std::pair`.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 5.973 `std::tuple_element< 1, pair< _Tp1, _Tp2 > >` Struct Template Reference

```
#include <stl_pair.h>
```

### Public Types

- `typedef _Tp2 type`

#### 5.973.1 Detailed Description

```
template<class _Tp1, class _Tp2>  
struct std::tuple_element< 1, pair< _Tp1, _Tp2 > >
```

Partial specialization for `std::pair`.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 5.974 `std::tuple_element< __i, tuple< _Types... > >` Struct Template Reference

```
#include <tuple>
```

### Public Types

- `using type`

#### 5.974.1 Detailed Description

```
template<size_t __i, typename... _Types>  
struct std::tuple_element< __i, tuple< _Types... > >
```

Trait to get the *l*th element type from a tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.975 `std::tuple_element< _Ind, array< _Tp, _Nm > >` Struct Template Reference

```
#include <array>
```

## Public Types

- using **type**

### 5.975.1 Detailed Description

```
template<size_t _Ind, typename _Tp, size_t _Nm>
struct std::tuple_element< _Ind, array< _Tp, _Nm > >
```

Partial specialization for `std::array`.

The documentation for this struct was generated from the following file:

- [array](#)

### 5.976 `std::tuple_size< _Tp >` Struct Template Reference

Inherited by `std::tuple_size< const __enable_if_has_tuple_size< _Tp > >`, `std::tuple_size< const volatile __enable_if_has_tuple_size< _Tp > >`, and `std::tuple_size< volatile __enable_if_has_tuple_size< _Tp > >`.

#### 5.976.1 Detailed Description

```
template<typename _Tp>
struct std::tuple_size< _Tp >
```

Finds the size of a given tuple type.

The documentation for this struct was generated from the following file:

- [utility.h](#)

### 5.977 `std::tuple_size< array< _Tp, _Nm > >` Struct Template Reference

```
#include <array>
```

Inheritance diagram for `std::tuple_size< array< _Tp, _Nm > >`:



## Public Types

- using **type**
- using **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** () const noexcept

**Static Public Attributes**

- static constexpr size\_t **value**

**5.977.1 Detailed Description**

```
template<typename _Tp, size_t _Nm>
struct std::tuple_size< array< _Tp, _Nm > >
```

Partial specialization for `std::array`.

The documentation for this struct was generated from the following file:

- [array](#)

**5.978 `std::tuple_size< pair< _Tp1, _Tp2 > >` Struct Template Reference**

```
#include <stl_pair.h>
```

Inheritance diagram for `std::tuple_size< pair< _Tp1, _Tp2 > >`:

**Public Types**

- using **type**
- using **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** () const noexcept

**Static Public Attributes**

- static constexpr size\_t **value**

### 5.978.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_size< pair< _Tp1, _Tp2 > >
```

Partial specialization for `std::pair`.

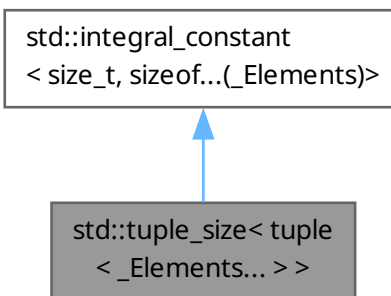
The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

### 5.979 std::tuple\_size< tuple< \_Elements... > > Struct Template Reference

```
#include <tuple>
```

Inheritance diagram for `std::tuple_size< tuple< _Elements... > >`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr **size\_t value**

### 5.979.1 Detailed Description

```
template<typename... _Elements>
struct std::tuple_size< tuple< _Elements... > >
```

class `tuple_size`

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.980 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>::type` Struct Reference

```
#include <entry_cmp.hpp>
```

### Public Member Functions

- **type** (const Cmp\_Fn &other)
- bool **operator()** (entry lhs, entry rhs) const

#### 5.980.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>::type
```

Compare plus entry.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 5.981 `std::type_index` Struct Reference

```
#include <typeindex>
```

### Public Member Functions

- **type\_index** (const [type\\_info](#) &\_\_rhs) noexcept
- **size\_t hash\_code** () const noexcept
- const char \* **name** () const noexcept
- bool **operator!=** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator<** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator<=** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator==** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator>** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator>=** (const [type\\_index](#) &\_\_rhs) const noexcept

#### 5.981.1 Detailed Description

Class `type_index`.

The class `type_index` provides a simple wrapper for `type_info` which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

The documentation for this struct was generated from the following file:

- [typeindex](#)

## 5.982 `__gnu_debug::type_info` Class Reference

```
#include <typeinfo>
```

### Public Member Functions

- virtual [~type\\_info](#) ()
- virtual bool **\_\_do\_catch** (const [type\\_info](#) \*\_\_thr\_type, void \*\*\_\_thr\_obj, unsigned \_\_outer) const
- virtual bool **\_\_do\_upcast** (const `__cxxabiv1::__class_type_info` \*\_\_target, void \*\*\_\_obj\_ptr) const
- virtual bool **\_\_is\_function\_p** () const

- virtual bool `__is_pointer_p` () const
- bool `before` (const `type_info` &\_\_arg) const noexcept
- size\_t `hash_code` () const noexcept
- const char \* `name` () const noexcept
- constexpr bool `operator==` (const `type_info` &\_\_arg) const noexcept

### Protected Member Functions

- `type_info` (const char \*\_\_n)

### Protected Attributes

- const char \* `__name`

## 5.982.1 Detailed Description

Part of RTTI.

The `type_info` class describes type information generated by an implementation.

## 5.982.2 Constructor & Destructor Documentation

### `~type_info()`

```
virtual std::type_info::~~type_info () [virtual]
```

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated `type_info` structures in the new-abi.

## 5.982.3 Member Function Documentation

### `before()`

```
bool std::type_info::before (  
    const type_info & __arg) const [noexcept]
```

Returns true if \*this precedes \_\_arg in the implementation's collation order.

### `name()`

```
const char * std::type_info::name () const [inline], [noexcept]
```

Returns an *implementation-defined* byte string; this is not portable between compilers!

The documentation for this class was generated from the following file:

- `typeinfo`

## 5.983 std::type\_info Class Reference

```
#include <typeinfo>
```

### Public Member Functions

- virtual `~type_info` ()
- virtual bool `__do_catch` (const `type_info` \*\_\_thr\_type, void \*\*\_\_thr\_obj, unsigned \_\_outer) const
- virtual bool `__do_upcast` (const `__cxxabiv1::__class_type_info` \*\_\_target, void \*\*\_\_obj\_ptr) const
- virtual bool `__is_function_p` () const
- virtual bool `__is_pointer_p` () const
- bool `before` (const `type_info` &\_\_arg) const noexcept
- size\_t `hash_code` () const noexcept

- `const char * name () const noexcept`
- `constexpr bool operator== (const type\_info &__arg) const noexcept`

#### Protected Member Functions

- `type_info (const char * __n)`

#### Protected Attributes

- `const char * __name`

### 5.983.1 Detailed Description

Part of RTTI.

The `type_info` class describes type information generated by an implementation.

### 5.983.2 Constructor & Destructor Documentation

#### `~type_info()`

```
virtual std::type_info::~~type_info () [virtual]
```

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated `type_info` structures in the new-abi.

### 5.983.3 Member Function Documentation

#### `before()`

```
bool std::type_info::before (
    const type\_info & __arg) const [noexcept]
```

Returns true if `*this` precedes `__arg` in the implementation's collation order.

#### `name()`

```
const char * std::type_info::name () const [inline], [noexcept]
```

Returns an *implementation-defined* byte string; this is not portable between compilers!

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.984 `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

```
#include <types_traits.hpp>
```



Inheritance diagram for `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`:



## Public Types

- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `__rebind_ka::const_pointer` **key\_const\_pointer**
- typedef `__rebind_ka::const_reference` **key\_const\_reference**
- typedef `__rebind_ka::pointer` **key\_pointer**
- typedef `__rebind_ka::reference` **key\_reference**
- typedef `Key` **key\_type**
- typedef `__rebind_ma::const_pointer` **mapped\_const\_pointer**
- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `Mapped` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `select_value_type< Key, Mapped >::type` **value\_type**

## Public Attributes

- `no_throw_indicator` **m\_no\_throw\_copies\_indicator**
- `store_extra` **m\_store\_extra\_indicator**

### 5.984.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >
```

Traits for abstract types.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 5.985 std::chrono::tzdb\_list Class Reference

```
#include <chrono>
```

### Classes

- class [const\\_iterator](#)

### Public Member Functions

- **tzdb\_list** (const [tzdb\\_list](#) &)=delete
- [const\\_iterator](#) **begin** () const noexcept
- [const\\_iterator](#) **cbegin** () const noexcept
- [const\\_iterator](#) **cend** () const noexcept
- [const\\_iterator](#) **end** () const noexcept
- [const\\_iterator](#) **erase\_after** ([const\\_iterator](#) \_\_p)
- const tzdb & **front** () const noexcept
- [tzdb\\_list](#) & **operator=** (const [tzdb\\_list](#) &)=delete

### Friends

- const tzdb & **get\_tzdb** ()
- [tzdb\\_list](#) & **get\_tzdb\_list** ()
- class **leap\_second**
- const tzdb & **reload\_tzdb** ()
- struct **time\_zone::\_Impl**
- class **time\_zone\_link**
- struct **tzdb**

### 5.985.1 Detailed Description

The list of `chrono::tzdb` objects

A single object of this type is constructed by the C++ runtime, and can be accessed by calling `chrono::get_tzdb_list()`.

The front of the list is the current `tzdb` object and can be accessed via `chrono::get_tzdb_list().front()` or `chrono::get_tzdb()` or `*chronoget_tzdb_list().begin()`.

The `chrono::reload_tzdb()` function will check for a newer version and if found, insert it at the front of the list.

Since

C++20

### 5.985.2 Member Function Documentation

#### erase\_after()

```
const_iterator std::chrono::tzdb_list::erase_after (
    const_iterator __p)
```

Remove the tzdb object *after* the one the iterator refers to.

Calling this function concurrently with any of `front()`, `begin()`, or `end()` does not cause a data race, but in general this function is not thread-safe. The behaviour may be undefined if erasing an element from the list while another thread is calling the same function, or incrementing an iterator into the list, or accessing the element being erased (unless it is accessed through an iterator).

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A dereferenceable iterator. |
|------------------|-----------------------------|

#### Returns

An iterator the element after the one that was erased (or `end()` if there is no such element).

#### Since

C++20

#### front()

```
const tzdb & std::chrono::tzdb_list::front () const [noexcept]
```

Access the current tzdb at the front of the list.

This returns a reference to the same object as `chrono::get_tzdb()`.

#### Returns

A reference to the current tzdb object.

#### Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

### 5.986 `__gnu_cxx::unary_compose<_Operation1, _Operation2 >` Class Template Reference

```
#include <functional>
```

Inheritance diagram for `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`:



### Public Types

- typedef `_Operation2::argument_type` [argument\\_type](#)
- typedef `_Operation1::result_type` [result\\_type](#)

### Public Member Functions

- **unary\_compose** (const `_Operation1` &\_\_x, const `_Operation2` &\_\_y)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &\_\_x) const

### Protected Attributes

- `_Operation1` **\_M\_fn1**
- `_Operation2` **\_M\_fn2**

#### 5.986.1 Detailed Description

```
template<class _Operation1, class _Operation2>
class __gnu_cxx::unary_compose< _Operation1, _Operation2 >
```

An [SGI extension](#) .

#### 5.986.2 Member Typedef Documentation

##### **argument\_type**

```
typedef _Operation2::argument_type std::unary_function< _Operation2::argument_type, _Operation1↵
::result_type >::argument_type [inherited]
argument_type is the type of the argument
```

##### **result\_type**

```
typedef _Operation1::result_type std::unary_function< _Operation2::argument_type, _Operation1↵
::result_type >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.987 std::unary\_function< \_Arg, \_Result > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::unary\_function< \_Arg, \_Result >:



### Public Types

- typedef \_Arg [argument\\_type](#)
- typedef \_Result [result\\_type](#)

#### 5.987.1 Detailed Description

```
template<typename _Arg, typename _Result>
struct std::unary_function< _Arg, _Result >
```

Helper for defining adaptable unary function objects.

**Deprecated** Deprecated in C++11, no longer in the standard since C++17.

#### 5.987.2 Member Typedef Documentation

##### argument\_type

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary\_function< \_Arg, \_Result >::argument\_type
argument_type is the type of the argument
```

##### result\_type

```
template<typename _Arg, typename _Result>
typedef _Result std::unary\_function< \_Arg, \_Result >::result\_type
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.988 std::unary\_negate< \_Predicate > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::unary\_negate< \_Predicate >:



### Public Types

- typedef `_Predicate::argument_type` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

### Public Member Functions

- constexpr **unary\_negate** (const `_Predicate` &\_\_x)
- constexpr `bool` **operator()** (const typename `_Predicate::argument_type` &\_\_x) const

### Protected Attributes

- `_Predicate` **M\_pred**

#### 5.988.1 Detailed Description

```
template<typename _Predicate>
class std::unary_negate< _Predicate >
```

One of the [negation functors](#).

#### 5.988.2 Member Typedef Documentation

##### **argument\_type**

```
typedef _Predicate::argument_type std::unary\_function< _Predicate::argument_type, bool >::argument_type [inherited]
```

`argument_type` is the type of the argument

**result\_type**

```
typedef bool std::unary\_function< \_Predicate::argument\_type, bool >::result_type [inherited]
```

result\_type is the return type

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**5.989 \_\_gnu\_parallel::unbalanced\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:

**Public Member Functions**

- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

**5.989.1 Detailed Description**

Recommends parallel execution using static load-balancing at compile time.

**5.989.2 Member Function Documentation****`__get_num_threads()`**

```
\_ThreadIndex \_\_gnu\_parallel::parallel\_tag::\_\_get\_num\_threads () [inline], [inherited]
```

Find out desired number of threads.

**Returns**

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#)

**`set_num_threads()`**

```
void \_\_gnu\_parallel::parallel\_tag::set\_num\_threads (  
    \_ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

## Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.990 `std::underflow_error` Class Reference

```
#include <stdexcept>
```

Inheritance diagram for `std::underflow_error`:



### Public Member Functions

- `underflow_error` (const char \*)
- `underflow_error` (const [string](#) &\_\_arg)
- `underflow_error` (const [underflow\\_error](#) &)=default
- `underflow_error` ([underflow\\_error](#) &&)=default
- `underflow_error` & **operator=** (const [underflow\\_error](#) &)=default
- `underflow_error` & **operator=** ([underflow\\_error](#) &&)=default
- virtual const char \* [what](#) () const noexcept

#### 5.990.1 Detailed Description

Thrown to indicate arithmetic underflow.

#### 5.990.2 Member Function Documentation

##### `what()`

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).



Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), and [std::filesystem::filesystem\\_error](#).  
 The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.991 std::underlying\_type<\_Tp> Struct Template Reference

```
#include <type_traits>
```

### 5.991.1 Detailed Description

```
template<typename _Tp>
struct std::underlying_type<_Tp>
```

The underlying type of an enum.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.992 std::uniform\_int\_distribution<\_IntType> Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_IntType [result\\_type](#)

### Public Member Functions

- [uniform\\_int\\_distribution](#) ()
- [uniform\\_int\\_distribution](#) (\_IntType \_\_a, \_IntType \_\_b=[\\_\\_gnu\\_cxx::\\_\\_int\\_traits](#)<\_IntType>::\_\_max)
- [uniform\\_int\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomBitGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomBitGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomBitGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomBitGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomBitGenerator>  
void [\\_\\_generate](#) ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomBitGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) [a](#) () const
- [result\\_type](#) [b](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename \_UniformRandomBitGenerator>  
[result\\_type](#) [operator\(\)](#) (\_UniformRandomBitGenerator &\_\_urng)
- template<typename \_UniformRandomBitGenerator>  
[result\\_type](#) [operator\(\)](#) (\_UniformRandomBitGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

**Friends**

- bool `operator==` (const `uniform_int_distribution` &\_\_d1, const `uniform_int_distribution` &\_\_d2)

**5.992.1 Detailed Description**

```
template<typename _IntType = int>
class std::uniform_int_distribution< _IntType >
```

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

Since

C++11

**5.992.2 Member Typedef Documentation****result\_type**

```
template<typename _IntType = int>
typedef _IntType std::uniform_int_distribution< _IntType >::result_type
```

The type of the range of the distribution.

**5.992.3 Constructor & Destructor Documentation****uniform\_int\_distribution() [1/2]**

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution () [inline]
```

Constructs a uniform distribution object.

**uniform\_int\_distribution() [2/2]**

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution (
    _IntType __a,
    _IntType __b = __gnu_cxx::__int_traits<_IntType>::__max) [inline], [explicit]
```

Constructs a uniform distribution object.

**5.992.4 Member Function Documentation****max()**

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::max () const [inline]
```

Returns the inclusive upper bound of the distribution range.

**min()**

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::min () const [inline]
```

Returns the inclusive lower bound of the distribution range.

**operator>()()**

```
template<typename _IntType = int>
template<typename _UniformRandomBitGenerator>
```

```
result_type std::uniform_int_distribution< _IntType >::operator() (
    _UniformRandomBitGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)](#).

#### **param()** [1/2]

```
template<typename _IntType = int>
param_type std::uniform_int_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

#### **param()** [2/2]

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

#### **reset()**

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

Does nothing for the uniform integer distribution.

### 5.992.5 Friends And Related Symbol Documentation

#### **operator==**

```
template<typename _IntType = int>
bool operator== (
    const uniform_int_distribution< _IntType > & __d1,
    const uniform_int_distribution< _IntType > & __d2) [friend]
```

Return true if two uniform integer distributions have the same parameters.

The documentation for this class was generated from the following file:

- [uniform\\_int\\_dist.h](#)

## 5.993 std::uniform\_real\_distribution< \_RealType > Class Template Reference

```
#include <random>
```

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [uniform\\_real\\_distribution](#) ()
- [uniform\\_real\\_distribution](#) (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- [uniform\\_real\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [template](#)<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- [template](#)<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [template](#)<typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) [a](#) () const
- [result\\_type](#) [b](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- [template](#)<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator](#)() (\_UniformRandomNumberGenerator &\_\_urng)
- [template](#)<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator](#)() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- bool [operator==](#) (const [uniform\\_real\\_distribution](#) &\_\_d1, const [uniform\\_real\\_distribution](#) &\_\_d2)

### 5.993.1 Detailed Description

```
template<typename _RealType = double>
class std::uniform_real_distribution<_RealType>
```

Uniform continuous distribution for random numbers.

A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Since

C++11

### 5.993.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::uniform_real_distribution<_RealType>::result_type
```

The type of the range of the distribution.

### 5.993.3 Constructor & Destructor Documentation

#### uniform\_real\_distribution() [1/2]

```
template<typename _RealType = double>
std::uniform_real_distribution<_RealType>::uniform_real_distribution () [inline]
```

Constructs a uniform\_real\_distribution object.

The lower bound is set to 0.0 and the upper bound to 1.0

**uniform\_real\_distribution()** [2/2]

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution (
    _RealType __a,
    _RealType __b = _RealType(1)) [inline], [explicit]
```

Constructs a `uniform_real_distribution` object.

**Parameters**

|                  |                                           |
|------------------|-------------------------------------------|
| <code>__a</code> | [IN] The lower bound of the distribution. |
| <code>__b</code> | [IN] The upper bound of the distribution. |

**5.993.4 Member Function Documentation****max()**

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::max () const [inline]
```

Returns the inclusive upper bound of the distribution range.

**min()**

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::min () const [inline]
```

Returns the inclusive lower bound of the distribution range.

**operator()()**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::uniform_real_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by `operator()()`.

**param()** [1/2]

```
template<typename _RealType = double>
param_type std::uniform_real_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by `std::operator>>()`.

**param()** [2/2]

```
template<typename _RealType = double>
void std::uniform_real_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::uniform\_real\_distribution<_RealType>::reset () [inline]
```

Resets the distribution state.  
Does nothing for the uniform real distribution.

**5.993.5 Friends And Related Symbol Documentation****operator==**

```
template<typename _RealType = double>
bool operator== (
    const uniform\_real\_distribution<_RealType> & __d1,
    const uniform\_real\_distribution<_RealType> & __d2) [friend]
```

Return true if two uniform real distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.994 `std::unique_lock<_Mutex>` Class Template Reference**

```
#include <mutex>
```

**Public Types**

- typedef `_Mutex` **mutex\_type**

**Public Member Functions**

- **unique\_lock** (const [unique\\_lock](#) &)=delete
- **unique\_lock** (mutex\_type &\_\_m)
- **unique\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#)) noexcept
- template<typename \_Rep, typename \_Period>  
  **unique\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)<\_Rep, \_Period> &\_\_rtime)
- template<typename \_Clock, typename \_Duration>  
  **unique\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)<\_Clock, \_Duration> &\_\_atime)
- **unique\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#)) noexcept
- **unique\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **unique\_lock** ([unique\\_lock](#) &&\_\_u) noexcept
- void **lock** ()
- mutex\_type \* **mutex** () const noexcept
- **operator bool** () const noexcept
- [unique\\_lock](#) & **operator=** (const [unique\\_lock](#) &)=delete
- [unique\\_lock](#) & **operator=** ([unique\\_lock](#) &&\_\_u) noexcept
- bool **owns\_lock** () const noexcept
- mutex\_type \* **release** () noexcept
- void **swap** ([unique\\_lock](#) &\_\_u) noexcept
- bool **try\_lock** ()

- `template<typename _Rep, typename _Period>`  
`bool try_lock_for (const chrono::duration< _Rep, _Period > &__rtime)`
- `template<typename _Clock, typename _Duration>`  
`bool try_lock_until (const chrono::time_point< _Clock, _Duration > &__atime)`
- `void unlock ()`

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _Mutex>`  
`void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept`

### 5.994.1 Detailed Description

**template<typename \_Mutex>**  
**class std::unique\_lock< \_Mutex >**

A movable scoped lock type.

A `unique_lock` controls mutex ownership within a scope. Ownership of the mutex can be delayed until after construction and can be transferred to another `unique_lock` by move construction or move assignment. If a mutex lock is owned when the destructor runs ownership will be released.

Since

C++11

### 5.994.2 Friends And Related Symbol Documentation

#### swap()

```
template<typename _Mutex>
void swap (
    unique_lock< _Mutex > & __x,
    unique_lock< _Mutex > & __y) [related]
```

Swap overload for `unique_lock` objects.

The documentation for this class was generated from the following file:

- [unique\\_lock.h](#)

## 5.995 std::unique\_ptr< \_Tp, \_Dp > Class Template Reference

```
#include <memory>
```

### Public Types

- using **deleter\_type**
- using **element\_type**
- using **pointer**

### Public Member Functions

- `template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>`  
`constexpr unique_ptr () noexcept`
- `template<typename _Up, typename = _Require<is_convertible<_Up*, pointer>, is_same<_Dp, default_delete<_Tp>>>>`  
`unique_ptr (auto_ptr< _Up > &&__u) noexcept`
- `unique_ptr (const unique_ptr &)=delete`

- template<typename \_Del = \_Dp, typename = \_DeleterConstraint<\_Del>>>  
constexpr [unique\\_ptr](#) (nullptr\_t) noexcept
- template<typename \_Del = \_Dp, typename = \_DeleterConstraint<\_Del>>>  
constexpr [unique\\_ptr](#) (pointer \_\_p) noexcept
- template<typename \_Del = deleter\_type, typename = \_Require<is\_move\_constructible<\_Del>>>>  
constexpr [unique\\_ptr](#) (pointer \_\_p, \_\_enable\_if\_t<!is\_lvalue\_reference<\_Del>::value, \_Del && > \_\_d) noexcept
- template<typename \_Del = deleter\_type, typename = \_Require<is\_copy\_constructible<\_Del>>>>  
constexpr [unique\\_ptr](#) (pointer \_\_p, const deleter\_type &\_\_d) noexcept
- template<typename \_Del = deleter\_type, typename \_DelUnref = typename remove\_reference<\_Del>::type>  
constexpr [unique\\_ptr](#) (pointer, \_\_enable\_if\_t<is\_lvalue\_reference<\_Del>::value, \_DelUnref && >)=delete
- [unique\\_ptr](#) ([unique\\_ptr](#) &&)=default
- template<typename \_Up, typename \_Ep, typename = \_Require<\_\_safe\_conversion\_up<\_Up, \_Ep>, \_\_conditional\_t<is\_reference<\_Up>::value, is\_same<\_Ep, \_Dp>, is\_convertible<\_Ep, \_Dp>>>>>  
constexpr [unique\\_ptr](#) ([unique\\_ptr](#) < \_Up, \_Ep > &&\_\_u) noexcept
- ~[unique\\_ptr](#) () noexcept
- constexpr pointer [get](#) () const noexcept
- constexpr const deleter\_type & [get\\_deleter](#) () const noexcept
- constexpr deleter\_type & [get\\_deleter](#) () noexcept
- constexpr [operator bool](#) () const noexcept
- constexpr [add\\_lvalue\\_reference](#)< element\_type >::type [operator\\*](#) () const noexcept(noexcept(\*std::declval<pointer >()))
- constexpr pointer [operator->](#) () const noexcept
- [unique\\_ptr](#) & [operator=](#) (const [unique\\_ptr](#) &)=delete
- constexpr [unique\\_ptr](#) & [operator=](#) (nullptr\_t) noexcept
- [unique\\_ptr](#) & [operator=](#) ([unique\\_ptr](#) &&)=default
- template<typename \_Up, typename \_Ep>  
constexpr [enable\\_if](#)<\_\_and<\_\_safe\_conversion\_up<\_Up, \_Ep>, is\_assignable< deleter\_type &, \_Ep && >::value, [unique\\_ptr](#) & >::type [operator=](#) ([unique\\_ptr](#) < \_Up, \_Ep > &&\_\_u) noexcept
- constexpr pointer [release](#) () noexcept
- constexpr void [reset](#) (pointer \_\_p=pointer()) noexcept
- constexpr void [swap](#) ([unique\\_ptr](#) &\_\_u) noexcept

## Related Symbols

(Note that these are not member symbols.)

- template<typename \_CharT, typename \_Traits, typename \_Tp, typename \_Dp>  
[basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [unique\\_ptr](#)< \_Tp, \_Dp > &\_\_p)
- template<typename \_Tp, typename \_Dp>  
constexpr [enable\\_if](#)<\_\_is\_swappable<\_Dp>::value >::type [swap](#) ([unique\\_ptr](#)< \_Tp, \_Dp > &\_\_x, [unique\\_ptr](#)< \_Tp, \_Dp > &\_\_y) noexcept

### 5.995.1 Detailed Description

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
class std::unique_ptr< _Tp, _Dp >
```

A move-only smart pointer that manages unique ownership of a resource.

Since

C++11



## 5.995.2 Constructor & Destructor Documentation

### unique\_ptr() [1/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr () [inline], [constexpr], [noexcept]
```

Default constructor, creates a unique\_ptr that owns nothing.

### unique\_ptr() [2/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    pointer __p) [inline], [explicit], [constexpr], [noexcept]
```

Takes ownership of a pointer.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| $\leftarrow$<br>_p | A pointer to an object of element_type |
|--------------------|----------------------------------------|

The deleter will be value-initialized.

### unique\_ptr() [3/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_copy_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    pointer __p,
    const deleter_type & __d) [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| $\leftarrow$<br>_p | A pointer to an object of element_type |
| $\leftarrow$<br>_d | A reference to a deleter.              |

The deleter will be initialized with \_\_d

### unique\_ptr() [4/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_move_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    pointer __p,
    __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) [inline], [constexpr],
[noexcept]
```

Takes ownership of a pointer.

## Parameters

|                    |                                                   |
|--------------------|---------------------------------------------------|
| $\leftarrow$<br>_p | A pointer to an object of element_type            |
| $\leftarrow$<br>_d | An rvalue reference to a (non-reference) deleter. |

The deleter will be initialized with `std::move(__d)`

**unique\_ptr()** [5/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    nullptr_t ) [inline], [constexpr], [noexcept]
```

Creates a unique\_ptr that owns nothing.

**unique\_ptr()** [6/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    unique_ptr< _Tp, _Dp > && ) [default]
```

Move constructor.

**unique\_ptr()** [7/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __
conditional_t<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    unique_ptr< _Up, _Ep > && __u) [inline], [constexpr], [noexcept]
```

Converting constructor from another type.

Requires that the pointer owned by \_\_u is convertible to the type of pointer owned by this object, \_\_u does not own an array, and \_\_u has a compatible deleter type.

**unique\_ptr()** [8/8]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    auto_ptr< _Up > && __u) [inline], [noexcept]
```

Converting constructor from auto\_ptr.

**~unique\_ptr()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::~~unique_ptr () [inline], [noexcept]
```

Destructor, invokes the deleter if the stored pointer is not null.

**5.995.3 Member Function Documentation****get()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::get () const [inline], [constexpr], [noexcept]
```

Return the stored pointer.

Referenced by [std::unique\\_ptr<\\_State>::operator bool\(\)](#), [std::unique\\_ptr<\\_State>::operator\\*\(\)](#), [std::unique\\_ptr<\\_State>::operator->\(\)](#), [std::operator<\(\)>](#), [std::operator<\(\)>](#), [std::operator<\(\)>](#), [std::unique\\_ptr<\\_State>::operator<<\(\)>](#), [std::operator==\(\(\), std::operator>\(\)\)](#), and [std::operator>\(\)](#).

### **get\_deleter() [1/2]**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
const deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter () const [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.

### **get\_deleter() [2/2]**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter () [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.

Referenced by [std::unique\\_ptr<\\_State>::~~unique\\_ptr\(\)](#), and [std::unique\\_ptr<\\_State>::operator=\(\)](#).

### **operator bool()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::operator bool () const [inline], [explicit], [constexpr], [noexcept]
```

Return true if the stored pointer is not null.

### **operator\*()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
add_lvalue_reference< element_type >::type std::unique_ptr< _Tp, _Dp >::operator* () const [inline],
[constexpr], [noexcept]
```

Dereference the stored pointer.

### **operator->()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::operator-> () const [inline], [constexpr], [noexcept]
```

Return the stored pointer.

### **operator=() [1/3]**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (
    nullptr_t ) [inline], [constexpr], [noexcept]
```

Reset the unique\_ptr to empty, invoking the deleter if necessary.

### **operator=() [2/3]**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (
    unique_ptr< _Tp, _Dp > && ) [default]
```

Move assignment operator.

Invokes the deleter if this object owns a pointer.

**operator=()** [3/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep>
enable_if< __and< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && >
>::value, unique_ptr & >::type std::unique_ptr< _Tp, _Dp >::operator= (
    unique_ptr< _Up, _Ep > && __u) [inline], [constexpr], [noexcept]
```

Assignment from another type.

**Parameters**

|                  |                                                                                                |
|------------------|------------------------------------------------------------------------------------------------|
| <code>__u</code> | The object to transfer ownership from, which owns a convertible pointer to a non-array object. |
|------------------|------------------------------------------------------------------------------------------------|

Invokes the deleter if this object owns a pointer.

**release()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::release () [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.

**reset()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr< _Tp, _Dp >::reset (
    pointer __p = pointer()) [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__p</code> | The new pointer to store. |
|------------------|---------------------------|

The deleter will be invoked if a pointer is already owned.

Referenced by `std::unique_ptr<_State>::operator=()`, and `std::unique_ptr<_State>::operator=()`.

**swap()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr< _Tp, _Dp >::swap (
    unique_ptr< _Tp, _Dp > & __u) [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.

The documentation for this class was generated from the following files:

- [unique\\_ptr.h](#)
- [auto\\_ptr.h](#)

**5.996 `std::unique_ptr<_Tp[],_Dp>` Class Template Reference**

```
#include <memory>
```

## Public Types

- `template<typename _Up>`  
    `using __safe_conversion_raw`
- `template<typename _Up, typename _Ep, typename _UPtr = unique_ptr<_Up, _Ep>, typename _UP_pointer = typename _UPtr::pointer,`  
    `typename _UP_element_type = typename _UPtr::element_type>`  
    `using __safe_conversion_up`
- `using deleter_type`
- `using deleter_type`
- `using element_type`
- `using element_type`
- `using pointer`
- `using pointer`

## Public Member Functions

- `constexpr unique_ptr () noexcept`
- `template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>`  
    `constexpr unique_ptr () noexcept`
- `template<typename _Up, typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename enable_if< __safe_↔`  
    `conversion_raw<_Up>::value, bool>::type>`  
    `constexpr unique_ptr (_Up __p) noexcept`
- `template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<_Up>, is_move_↔`  
    `constructible<_Del>>>`  
    `constexpr unique_ptr (_Up __p, __enable_if_t<!is_lvalue_reference<_Del>::value, _Del && > __d) noexcept`
- `template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<_Up>, is_copy_constructible<↔`  
    `_Del>>>`  
    `constexpr unique_ptr (_Up __p, const deleter_type &__d) noexcept`
- `template<typename _Up, typename _Del = deleter_type, typename _DelUnref = typename remove_reference<_Del>::type, typename =`  
    `_Require<__safe_conversion_raw<_Up>>>`  
    `unique_ptr (_Up, __enable_if_t< is_lvalue_reference<_Del>::value, _DelUnref && >)=delete`
- `unique_ptr (auto_ptr<_Up> &&__u) noexcept`
- `unique_ptr (const unique_ptr &)=delete`
- `unique_ptr (const unique_ptr &)=delete`
- `constexpr unique_ptr (nullptr_t) noexcept`
- `template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>`  
    `constexpr unique_ptr (nullptr_t) noexcept`
- `constexpr unique_ptr (pointer __p) noexcept`
- `constexpr unique_ptr (pointer __p, __enable_if_t<!is_lvalue_reference<_Del>::value, _Del && > __d) noex-`  
    `cept`
- `constexpr unique_ptr (pointer __p, const deleter_type &__d) noexcept`
- `constexpr unique_ptr (pointer, __enable_if_t< is_lvalue_reference<_Del>::value, _DelUnref && >)=delete`
- `unique_ptr (unique_ptr &&)=default`
- `unique_ptr (unique_ptr &&)=default`
- `constexpr unique_ptr (unique_ptr<_Up, _Ep> &&__u) noexcept`
- `template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __conditional_t<is_reference<↔`  
    `_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>`  
    `constexpr unique_ptr (unique_ptr<_Up, _Ep> &&__u) noexcept`
- `~unique_ptr ()`
- `~unique_ptr () noexcept`
- `constexpr pointer get () const noexcept`
- `constexpr pointer get () const noexcept`
- `constexpr const deleter_type & get_deleter () const noexcept`

- constexpr const deleter\_type & [get\\_deleter](#) () const noexcept
- constexpr deleter\_type & [get\\_deleter](#) () noexcept
- constexpr deleter\_type & [get\\_deleter](#) () noexcept
- constexpr [operator bool](#) () const noexcept
- constexpr [operator bool](#) () const noexcept
- constexpr [add\\_lvalue\\_reference](#)< element\_type >::type [operator\\*](#) () const noexcept(noexcept(\*std::declval< pointer >()))
- constexpr pointer [operator->](#) () const noexcept
- [unique\\_ptr](#) & [operator=](#) (const [unique\\_ptr](#) &)=delete
- [unique\\_ptr](#) & [operator=](#) (const [unique\\_ptr](#) &)=delete
- constexpr [unique\\_ptr](#) & [operator=](#) (nullptr\_t) noexcept
- constexpr [unique\\_ptr](#) & [operator=](#) (nullptr\_t) noexcept
- [unique\\_ptr](#) & [operator=](#) ([unique\\_ptr](#) &&)=default
- [unique\\_ptr](#) & [operator=](#) ([unique\\_ptr](#) &&)=default
- constexpr [enable\\_if](#)< \_\_and\_< \_\_safe\_conversion\_up< \_Up, \_Ep >, [is\\_assignable](#)< deleter\_type &, \_Ep && >::value, [unique\\_ptr](#) & >::type [operator=](#) ([unique\\_ptr](#)< \_Up, \_Ep > &&\_\_u) noexcept
- template<typename \_Up, typename \_Ep>  
constexpr [enable\\_if](#)< \_\_and\_< \_\_safe\_conversion\_up< \_Up, \_Ep >, [is\\_assignable](#)< deleter\_type &, \_Ep && >::value, [unique\\_ptr](#) & >::type [operator=](#) ([unique\\_ptr](#)< \_Up, \_Ep > &&\_\_u) noexcept
- constexpr [std::add\\_lvalue\\_reference](#)< element\_type >::type [operator\[\]](#) (size\_t \_\_i) const
- constexpr pointer [release](#) () noexcept
- constexpr pointer [release](#) () noexcept
- template<typename \_Up, typename = \_Require< \_\_or\_<is\_same<\_Up, pointer>, \_\_and\_<is\_same<pointer, element\_type\*>, is\_↔ pointer<\_Up>, is\_convertible< typename remove\_pointer<\_Up>::type(\*)[], element\_type(\*)[] > > >>  
constexpr void [reset](#) (\_Up \_\_p) noexcept
- constexpr void [reset](#) (nullptr\_t=nullptr) noexcept
- constexpr void [reset](#) (pointer \_\_p=pointer()) noexcept
- constexpr void [swap](#) ([unique\\_ptr](#) &\_\_u) noexcept
- constexpr void [swap](#) ([unique\\_ptr](#) &\_\_u) noexcept

## Related Symbols

(Note that these are not member symbols.)

- [basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [unique\\_ptr](#)< \_Tp, \_Dp > &\_\_p)
- [basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [unique\\_ptr](#)< \_Tp, \_Dp > &\_\_p)
- constexpr [enable\\_if](#)< \_\_is\_swappable< \_Dp >::value >::type [swap](#) ([unique\\_ptr](#)< \_Tp, \_Dp > &\_\_↔ x, [unique\\_ptr](#)< \_Tp, \_Dp > &\_\_y) noexcept
- constexpr [enable\\_if](#)< \_\_is\_swappable< \_Dp >::value >::type [swap](#) ([unique\\_ptr](#)< \_Tp, \_Dp > &\_\_↔ x, [unique\\_ptr](#)< \_Tp, \_Dp > &\_\_y) noexcept

### 5.996.1 Detailed Description

```
template<typename _Tp, typename _Dp>
class std::unique_ptr< _Tp[], _Dp >
```

A move-only smart pointer that manages unique ownership of an array.

Since

C++11

## 5.996.2 Constructor & Destructor Documentation

### unique\_ptr() [1/14]

```
template<typename _Tp, typename _Dp>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr () [inline], [constexpr], [noexcept]
```

Default constructor, creates a unique\_ptr that owns nothing.

References [unique\\_ptr\(\)](#).

Referenced by [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [~unique\\_ptr\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), and [swap\(\)](#).

### unique\_ptr() [2/14]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename
enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
    _Up __p) [inline], [explicit], [constexpr], [noexcept]
```

Takes ownership of a pointer.

#### Parameters

|                                                                                                                               |                                                                                             |
|-------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <a href="#"></a><br><a href="#"><u>p</u></a> | A pointer to an array of a type safely convertible to an array of <code>element_type</code> |
|-------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|

The deleter will be value-initialized.

References [unique\\_ptr\(\)](#).

### unique\_ptr() [3/14]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<  
_Up>, is_copy_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
    _Up __p,  
    const deleter_type & __d) [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

#### Parameters

|                                                                                                                                 |                                                                                             |
|---------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <a href="#"></a><br><a href="#"><u>p</u></a> | A pointer to an array of a type safely convertible to an array of <code>element_type</code> |
| <a href="#"></a><br><a href="#"><u>d</u></a> | A reference to a deleter.                                                                   |

The deleter will be initialized with `__d`

References [unique\\_ptr\(\)](#).

### unique\_ptr() [4/14]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<  
_Up>, is_move_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
```

```

    __Up __p,
    __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) [inline], [constexpr],
[noexcept]

```

Takes ownership of a pointer.

#### Parameters

|                  |                                                                                             |
|------------------|---------------------------------------------------------------------------------------------|
| <code>__p</code> | A pointer to an array of a type safely convertible to an array of <code>element_type</code> |
| <code>__d</code> | A reference to a deleter.                                                                   |

The deleter will be initialized with `std::move(__d)`

References [unique\\_ptr\(\)](#), and [std::move\(\)](#).

#### unique\_ptr() [5/14]

```

template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
    unique_ptr< _Tp[], _Dp > && ) [default]

```

Move constructor.

References [unique\\_ptr\(\)](#).

#### unique\_ptr() [6/14]

```

template<typename _Tp, typename _Dp>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
    nullptr_t ) [inline], [constexpr], [noexcept]

```

Creates a `unique_ptr` that owns nothing.

References [unique\\_ptr\(\)](#).

#### ~unique\_ptr() [1/2]

```

template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::~~unique_ptr () [inline]

```

Destructor, invokes the deleter if the stored pointer is not null.

References [unique\\_ptr\(\)](#), and [get\\_deleter\(\)](#).

#### unique\_ptr() [7/14]

```

std::unique_ptr< _Tp, _Dp >::unique_ptr () [inline], [constexpr], [noexcept]

```

Default constructor, creates a `unique_ptr` that owns nothing.

#### unique\_ptr() [8/14]

```

std::unique_ptr< _Tp, _Dp >::unique_ptr (
    pointer __p) [inline], [explicit], [constexpr], [noexcept]

```

Takes ownership of a pointer.

#### Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__p</code> | A pointer to an object of <code>element_type</code> |
|------------------|-----------------------------------------------------|

The deleter will be value-initialized.



**unique\_ptr()** [9/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    pointer __p,
    const deleter_type & __d) [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| $\leftarrow$<br>_p | A pointer to an object of element_type |
| $\leftarrow$<br>_d | A reference to a deleter.              |

The deleter will be initialized with \_\_d

**unique\_ptr()** [10/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    pointer __p,
    __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) [inline], [constexpr],
[noexcept]
```

Takes ownership of a pointer.

**Parameters**

|                    |                                                   |
|--------------------|---------------------------------------------------|
| $\leftarrow$<br>_p | A pointer to an object of element_type            |
| $\leftarrow$<br>_d | An rvalue reference to a (non-reference) deleter. |

The deleter will be initialized with `std::move(__d)`

**unique\_ptr()** [11/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    nullptr_t ) [inline], [constexpr], [noexcept]
```

Creates a unique\_ptr that owns nothing.

**unique\_ptr()** [12/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    unique_ptr< _Tp[], _Dp > && ) [default]
```

Move constructor.

**unique\_ptr()** [13/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    unique_ptr< _Up, _Ep > && __u) [inline], [constexpr], [noexcept]
```

Converting constructor from another type.

Requires that the pointer owned by \_\_u is convertible to the type of pointer owned by this object, \_\_u does not own an array, and \_\_u has a compatible deleter type.

**unique\_ptr()** [14/14]

```
std::unique_ptr<_Tp, _Dp>::unique_ptr (
    auto_ptr<_Up> && __u) [inline], [noexcept]
```

Converting constructor from `auto_ptr`.

**~unique\_ptr()** [2/2]

```
std::unique_ptr<_Tp, _Dp>::~~unique_ptr () [inline], [noexcept]
```

Destructor, invokes the deleter if the stored pointer is not null.

**5.996.3 Member Function Documentation****get()** [1/2]

```
pointer std::unique_ptr<_Tp, _Dp>::get () const [inline], [constexpr], [noexcept]
```

Return the stored pointer.

**get()** [2/2]

```
template<typename _Tp, typename _Dp>
pointer std::unique_ptr<_Tp[], _Dp>::get () const [inline], [constexpr], [noexcept]
```

Return the stored pointer.

References [get\(\)](#).

Referenced by [get\(\)](#), [operator bool\(\)](#), and [operator\[\]\(\)](#).

**get\_deleter()** [1/4]

```
const deleter_type & std::unique_ptr<_Tp, _Dp>::get_deleter () const [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.

**get\_deleter()** [2/4]

```
template<typename _Tp, typename _Dp>
const deleter_type & std::unique_ptr<_Tp[], _Dp>::get_deleter () const [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.

References [get\\_deleter\(\)](#).

**get\_deleter()** [3/4]

```
deleter_type & std::unique_ptr<_Tp, _Dp>::get_deleter () [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.

**get\_deleter()** [4/4]

```
template<typename _Tp, typename _Dp>
deleter_type & std::unique_ptr<_Tp[], _Dp>::get_deleter () [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.

References [get\\_deleter\(\)](#).

Referenced by [~unique\\_ptr\(\)](#), [get\\_deleter\(\)](#), [get\\_deleter\(\)](#), and [operator=\(\)](#).

**operator bool()** [1/2]

```
std::unique_ptr<_Tp, _Dp>::operator bool () const [inline], [explicit], [constexpr], [noexcept]
```

Return `true` if the stored pointer is not null.

**operator bool()** [2/2]

```
template<typename _Tp, typename _Dp>
```

```
std::unique_ptr< _Tp[], _Dp >::operator bool () const [inline], [explicit], [constexpr], [noexcept]
```

Return true if the stored pointer is not null.

References [get\(\)](#).

**operator\*()**

```
add_lvalue_reference< element_type >::type std::unique_ptr< _Tp, _Dp >::operator* () const [inline],  
[constexpr], [noexcept]
```

Dereference the stored pointer.

**operator->()**

```
pointer std::unique_ptr< _Tp, _Dp >::operator-> () const [inline], [constexpr], [noexcept]
```

Return the stored pointer.

**operator=()** [1/6]

```
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (  
    nullptr_t ) [inline], [constexpr], [noexcept]
```

Reset the unique\_ptr to empty, invoking the deleter if necessary.

**operator=()** [2/6]

```
template<typename _Tp, typename _Dp>
```

```
unique_ptr & std::unique_ptr< _Tp[], _Dp >::operator= (  
    nullptr_t ) [inline], [constexpr], [noexcept]
```

Reset the unique\_ptr to empty, invoking the deleter if necessary.

References [unique\\_ptr\(\)](#), [operator=\(\)](#), and [reset\(\)](#).

**operator=()** [3/6]

```
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (  
    unique_ptr< _Tp[], _Dp > && ) [default]
```

Move assignment operator.

Invokes the deleter if this object owns a pointer.

**operator=()** [4/6]

```
template<typename _Tp, typename _Dp>
```

```
unique_ptr & std::unique_ptr< _Tp[], _Dp >::operator= (  
    unique_ptr< _Tp[], _Dp > && ) [default]
```

Move assignment operator.

Invokes the deleter if this object owns a pointer.

References [unique\\_ptr\(\)](#), and [operator=\(\)](#).



Referenced by [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

**operator=()** [5/6]

```
enable_if< __and< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && >  
>::value, unique_ptr & >::type std::unique_ptr< _Tp, _Dp >::operator= (  
    unique_ptr< _Up, _Ep > && __u) [inline], [constexpr], [noexcept]
```

Assignment from another type.

## Parameters

|                                                                                                   |                                                                                                |
|---------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| <a href="#"></a> | The object to transfer ownership from, which owns a convertible pointer to a non-array object. |
| <a href="#"></a> |                                                                                                |



Invokes the deleter if this object owns a pointer.

**operator=()** [6/6]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Ep>
enable_if< __and< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && >::value, unique_ptr & >::type std::unique_ptr< _Tp[], _Dp >::operator= (
    unique_ptr< _Up, _Ep > && __u) [inline], [constexpr], [noexcept]
```

Assignment from another type.

## Parameters

|                                                                                                   |                                                                                             |
|---------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <a href="#"></a> | The object to transfer ownership from, which owns a convertible pointer to an array object. |
| <a href="#"></a> |                                                                                             |

Invokes the deleter if this object owns a pointer.

References [unique\\_ptr\(\)](#), [std::forward\(\)](#), [get\\_deleter\(\)](#), [operator=\(\)](#), and [reset\(\)](#).

**operator[]()**

```
template<typename _Tp, typename _Dp>
std::add_lvalue_reference< element_type >::type std::unique_ptr< _Tp[], _Dp >::operator[] (
    size_t __i) const [inline], [constexpr]
```

Access an element of owned array.

References [get\(\)](#), and [operator\[\]\(\)](#).

Referenced by [operator\[\]\(\)](#).

**release()** [1/2]

```
pointer std::unique_ptr< _Tp, _Dp >::release () [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.

**release()** [2/2]

```
template<typename _Tp, typename _Dp>
pointer std::unique_ptr< _Tp[], _Dp >::release () [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.

References [release\(\)](#).

Referenced by [release\(\)](#).

**reset()** [1/2]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename = _Require< __or<is_same<_Up, pointer>, __and<is_same<pointer,
element_type*>, is_pointer<_Up>, is_convertible< typename remove_pointer<_Up>::type(*)[], element←
_type(*)[] > > > >>
void std::unique_ptr< _Tp[], _Dp >::reset (
    _Up __p) [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

## Parameters

|                        |                           |
|------------------------|---------------------------|
| <code>_↵<br/>_p</code> | The new pointer to store. |
|------------------------|---------------------------|

The deleter will be invoked if a pointer is already owned.

References [std::move\(\)](#), and [reset\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), and [reset\(\)](#).

**reset()** [2/2]

```
void std::unique_ptr< _Tp, _Dp >::reset (
    pointer __p = pointer()) [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

## Parameters

|                        |                           |
|------------------------|---------------------------|
| <code>_↵<br/>_p</code> | The new pointer to store. |
|------------------------|---------------------------|

The deleter will be invoked if a pointer is already owned.

**swap()** [1/2]

```
void std::unique_ptr< _Tp, _Dp >::swap (
    unique_ptr< _Tp[], _Dp > & __u) [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.

**swap()** [2/2]

```
template<typename _Tp, typename _Dp>
void std::unique_ptr< _Tp[], _Dp >::swap (
    unique_ptr< _Tp[], _Dp > & __u) [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.

References [unique\\_ptr\(\)](#), and [swap\(\)](#).

Referenced by [swap\(\)](#).

**5.996.4 Friends And Related Symbol Documentation****operator<<()** [1/2]

```
basic_ostream< _CharT, _Traits > & operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const unique_ptr< _Tp, _Dp > & __p) [related]
```

Stream output operator for `unique_ptr`.

Since

C++20

**operator<<()** [2/2]

```
basic_ostream< _CharT, _Traits > & operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const unique_ptr< _Tp, _Dp > & __p) [related]
```

Stream output operator for `unique_ptr`.

Since

C++20

### swap() [1/2]

```
enable_if< __is_swappable< _Dp >::value >::type swap (
    unique_ptr< _Tp, _Dp > & __x,
    unique_ptr< _Tp, _Dp > & __y) [related]
```

Swap overload for unique\_ptr.

### swap() [2/2]

```
enable_if< __is_swappable< _Dp >::value >::type swap (
    unique_ptr< _Tp, _Dp > & __x,
    unique_ptr< _Tp, _Dp > & __y) [related]
```

Swap overload for unique\_ptr.

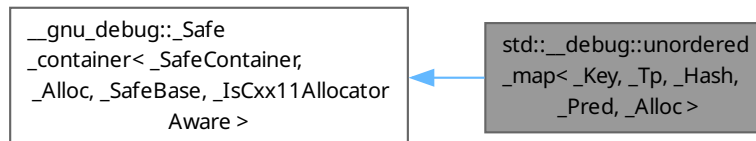
The documentation for this class was generated from the following files:

- [unique\\_ptr.h](#)
- [auto\\_ptr.h](#)

## 5.997 std::\_\_debug::unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_map>
```

Inheritance diagram for std::\_\_debug::unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >:



### Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_\_gnu\_debug::\_Safe\_iterator< \_Base\_const\_iterator, unordered\_map > **const\_iterator**
- typedef \_\_gnu\_debug::\_Safe\_local\_iterator< \_Base\_const\_local\_iterator, unordered\_map > **const\_local\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::hasher **hasher**
- using **insert\_return\_type**
- typedef \_\_gnu\_debug::\_Safe\_iterator< \_Base\_iterator, unordered\_map > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**

- typedef `__gnu_debug::_Safe_local_iterator`< `_Base_local_iterator`, `unordered_map` > `local_iterator`
- typedef `_Base::mapped_type` `mapped_type`
- using `node_type`
- typedef `_Base::pointer` `pointer`
- typedef `_Base::reference` `reference`
- typedef `_Base::size_type` `size_type`
- typedef `_Base::value_type` `value_type`

## Public Member Functions

- `unordered_map` (`_Base_ref __x`)
- template<typename `_InputIterator`>  
`unordered_map` (`_InputIterator __first`, `_InputIterator __last`, `const allocator_type &__a`)
- template<typename `_InputIterator`>  
`unordered_map` (`_InputIterator __first`, `_InputIterator __last`, `size_type __n`, `const allocator_type &__a`)
- template<typename `_InputIterator`>  
`unordered_map` (`_InputIterator __first`, `_InputIterator __last`, `size_type __n`, `const hasher &__hf`, `const allocator_type &__a`)
- template<typename `_InputIterator`>  
`unordered_map` (`_InputIterator __first`, `_InputIterator __last`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_map` (`const allocator_type &__a`)
- `unordered_map` (`const unordered_map &`)=default
- `unordered_map` (`const unordered_map &__umap`, `const allocator_type &__a`)
- `unordered_map` (`initializer_list`< `value_type` > `__l`, `const allocator_type &__a`)
- `unordered_map` (`initializer_list`< `value_type` > `__l`, `size_type __n`, `const allocator_type &__a`)
- `unordered_map` (`initializer_list`< `value_type` > `__l`, `size_type __n`, `const hasher &__hf`, `const allocator_type &__a`)
- `unordered_map` (`initializer_list`< `value_type` > `__l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_map` (`size_type __n`, `const allocator_type &__a`)
- `unordered_map` (`size_type __n`, `const hasher &__hf`, `const allocator_type &__a`)
- `unordered_map` (`size_type __n`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_map` (`unordered_map &&`)=default
- `unordered_map` (`unordered_map &&__umap`, `const allocator_type &__a`) noexcept(noexcept(`_Base::move`(`__umap`), `__a`)))
- `const _Base &_M_base` () const noexcept
- `_Base &_M_base` () noexcept
- `const_iterator begin` () const noexcept
- `iterator begin` () noexcept
- `local_iterator begin` (`size_type __b`)
- `const_local_iterator begin` (`size_type __b`) const
- `size_type bucket_size` (`size_type __b`) const
- `const_iterator cbegin` () const noexcept
- `const_local_iterator cbegin` (`size_type __b`) const
- `const_iterator cend` () const noexcept
- `const_local_iterator cend` (`size_type __b`) const
- void `clear` () noexcept
- template<typename... `_Args`>  
`std::pair`< `iterator`, `bool` > `emplace` (`_Args &&... __args`)

- `template<typename... _Args>`  
`iterator` **emplace\_hint** (`const_iterator` \_\_hint, `_Args` &&... \_\_args)
- `const_iterator` **end** () `const noexcept`
- `iterator` **end** () `noexcept`
- `local_iterator` **end** (`size_type` \_\_b)
- `const_local_iterator` **end** (`size_type` \_\_b) `const`
- `template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>`  
`std::pair< iterator, iterator >` **equal\_range** (`const _Kt` &\_\_k)
- `template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>`  
`std::pair< const_iterator, const_iterator >` **equal\_range** (`const _Kt` &\_\_k) `const`
- `std::pair< iterator, iterator >` **equal\_range** (`const key_type` &\_\_key)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (`const key_type` &\_\_key) `const`
- `_Base_iterator` **erase** (`_Base_const_iterator` \_\_it)
- `size_type` **erase** (`const key_type` &\_\_key)
- `iterator` **erase** (`const_iterator` \_\_first, `const_iterator` \_\_last)
- `iterator` **erase** (`const_iterator` \_\_it)
- `iterator` **erase** (`iterator` \_\_it)
- `node_type` **extract** (`const key_type` &\_\_key)
- `node_type` **extract** (`const_iterator` \_\_position)
- `template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>`  
`iterator` **find** (`const _Kt` &\_\_k)
- `template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>`  
`const_iterator` **find** (`const _Kt` &\_\_k) `const`
- `iterator` **find** (`const key_type` &\_\_key)
- `const_iterator` **find** (`const key_type` &\_\_key) `const`
- `template<typename _InputIterator>`  
`void` **insert** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`std::pair< iterator, bool >` **insert** (`_Pair` &&\_\_obj)
- `std::pair< iterator, bool >` **insert** (`const value_type` &\_\_obj)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator` **insert** (`const_iterator` \_\_hint, `_Pair` &&\_\_obj)
- `iterator` **insert** (`const_iterator` \_\_hint, `const value_type` &\_\_obj)
- `iterator` **insert** (`const_iterator` \_\_hint, `node_type` &&\_\_nh)
- `iterator` **insert** (`const_iterator` \_\_hint, `value_type` &&\_\_x)
- `insert_return_type` **insert** (`node_type` &&\_\_nh)
- `void` **insert** (`std::initializer_list< value_type >` \_\_l)
- `std::pair< iterator, bool >` **insert** (`value_type` &&\_\_x)
- `float` **max\_load\_factor** () `const noexcept`
- `void` **max\_load\_factor** (`float` \_\_f)
- `template<typename _H2, typename _P2>`  
`void` **merge** (`unordered_map< _Key, _Tp, _H2, _P2, _Alloc >` &&\_\_source)
- `template<typename _H2, typename _P2>`  
`void` **merge** (`unordered_map< _Key, _Tp, _H2, _P2, _Alloc >` &\_\_source)
- `template<typename _H2, typename _P2>`  
`void` **merge** (`unordered_multimap< _Key, _Tp, _H2, _P2, _Alloc >` &&\_\_source)
- `template<typename _H2, typename _P2>`  
`void` **merge** (`unordered_multimap< _Key, _Tp, _H2, _P2, _Alloc >` &\_\_source)
- `unordered_map` & **operator=** (`const unordered_map` &)=default
- `unordered_map` & **operator=** (`initializer_list< value_type >` \_\_l)
- `unordered_map` & **operator=** (`unordered_map` &)=default
- `void` **swap** (`unordered_map` &\_\_x) `noexcept(noexcept(declval<_Base &>().swap(__x)))`



### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug::\_Safe\_iterator
- template<typename \_ItT, typename \_SeqT>  
class ::\_\_gnu\_debug::\_Safe\_local\_iterator

#### 5.997.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_↵  
_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >>  
class std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class std::unordered\_map with safety/checking/debug instrumentation.  
The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

#### 5.998 std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_map>
```

### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::mapped\_type [mapped\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
  
- typedef \_Hashtable::pointer [pointer](#)
- typedef \_Hashtable::const\_pointer [const\\_pointer](#)
- typedef \_Hashtable::reference [reference](#)
- typedef \_Hashtable::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- [unordered\\_map](#) ()=default
- [template<typename \\_InputIterator>](#)  
[unordered\\_map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const [allocator\\_type](#) &\_\_a)
- [template<typename \\_InputIterator>](#)  
[unordered\\_map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- [template<typename \\_InputIterator>](#)  
[unordered\\_map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [template<typename \\_InputIterator>](#)  
[unordered\\_map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_map](#) (const [allocator\\_type](#) &\_\_a)
- [unordered\\_map](#) (const [unordered\\_map](#) &)=default
- [unordered\\_map](#) (const [unordered\\_map](#) &\_\_umap, const [allocator\\_type](#) &\_\_a)
- [unordered\\_map](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- [unordered\\_map](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- [unordered\\_map](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [unordered\\_map](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_map](#) ([size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- [unordered\\_map](#) ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [unordered\\_map](#) ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_map](#) ([unordered\\_map](#) &&)=default
- [unordered\\_map](#) ([unordered\\_map](#) &&\_\_umap, const [allocator\\_type](#) &\_\_a) noexcept(noexcept(\_Hashtable(std::move(\_\_umap.\_M\_h), \_\_a)))
- [iterator begin](#) () noexcept
- [local\\_iterator begin](#) ([size\\_type](#) \_\_n)
- [size\\_type bucket](#) (const [key\\_type](#) &\_\_key) const
- [size\\_type bucket\\_count](#) () const noexcept
- [size\\_type bucket\\_size](#) ([size\\_type](#) \_\_n) const
- [void clear](#) () noexcept
- [template<typename... \\_Args>](#)  
[std::pair](#)< [iterator](#), bool > [emplace](#) (\_Args &&... \_\_args)
- [template<typename... \\_Args>](#)  
[iterator emplace\\_hint](#) (const [iterator](#) \_\_pos, \_Args &&... \_\_args)
- [bool empty](#) () const noexcept
- [iterator end](#) () noexcept
- [local\\_iterator end](#) ([size\\_type](#) \_\_n)
- [size\\_type erase](#) (const [key\\_type](#) &\_\_x)
- [iterator erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [allocator\\_type get\\_allocator](#) () const noexcept
- [hasher hash\\_function](#) () const
- [template<typename \\_InputIterator>](#)  
[void insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [void insert](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [key\\_equal key\\_eq](#) () const
- [float load\\_factor](#) () const noexcept
- [size\\_type max\\_bucket\\_count](#) () const noexcept

- float `max_load_factor` () const noexcept
  - void `max_load_factor` (float \_\_z)
  - size\_type `max_size` () const noexcept
  - unordered\_map & `operator=` (const unordered\_map &)=default
  - unordered\_map & `operator=` (initializer\_list< value\_type > \_\_l)
  - unordered\_map & `operator=` (unordered\_map &&)=default
  - void `rehash` (size\_type \_\_n)
  - void `reserve` (size\_type \_\_n)
  - size\_type `size` () const noexcept
  - void `swap` (unordered\_map & \_\_x) noexcept(noexcept(\_M\_h.swap(\_\_x.\_M\_h)))
- 
- const\_iterator `begin` () const noexcept
  - const\_iterator `cbegin` () const noexcept
- 
- const\_iterator `end` () const noexcept
  - const\_iterator `cend` () const noexcept
- 
- std::pair< iterator, bool > `insert` (const value\_type & \_\_x)
  - std::pair< iterator, bool > `insert` (value\_type && \_\_x)
  - template<typename \_Pair>  
 \_\_enable\_if\_t< is\_constructible< value\_type, \_Pair && >::value, pair< iterator, bool > > `insert` (\_Pair && \_\_x)
- 
- iterator `insert` (const\_iterator \_\_hint, const value\_type & \_\_x)
  - iterator `insert` (const\_iterator \_\_hint, value\_type && \_\_x)
  - template<typename \_Pair>  
 \_\_enable\_if\_t< is\_constructible< value\_type, \_Pair && >::value, iterator > `insert` (const\_iterator \_\_hint, \_Pair && \_\_x)
- 
- iterator `erase` (const\_iterator \_\_position)
  - iterator `erase` (iterator \_\_position)
- 
- iterator `find` (const key\_type & \_\_x)
  - template<typename \_Kt>  
 auto `find` (const \_Kt & \_\_x) -> decltype(\_M\_h.\_M\_find\_tr(\_\_x))
  - const\_iterator `find` (const key\_type & \_\_x) const
  - template<typename \_Kt>  
 auto `find` (const \_Kt & \_\_x) const -> decltype(\_M\_h.\_M\_find\_tr(\_\_x))
- 
- size\_type `count` (const key\_type & \_\_x) const
  - template<typename \_Kt>  
 auto `count` (const \_Kt & \_\_x) const -> decltype(\_M\_h.\_M\_count\_tr(\_\_x))

- bool `contains` (const `key_type` &\_\_x) const
- template<typename \_Kt>  
auto `contains` (const \_Kt &\_\_x) const -> decltype(\_M\_h.\_M\_find\_tr(\_\_x), void(), true)
- std::pair< iterator, iterator > `equal_range` (const `key_type` &\_\_x)
- template<typename \_Kt>  
auto `equal_range` (const \_Kt &\_\_x) -> decltype(\_M\_h.\_M\_equal\_range\_tr(\_\_x))
- std::pair< const\_iterator, const\_iterator > `equal_range` (const `key_type` &\_\_x) const
- template<typename \_Kt>  
auto `equal_range` (const \_Kt &\_\_x) const -> decltype(\_M\_h.\_M\_equal\_range\_tr(\_\_x))
- `mapped_type` & `operator[]` (const `key_type` &\_\_k)
- `mapped_type` & `operator[]` (`key_type` &&\_\_k)
- `mapped_type` & `at` (const `key_type` &\_\_k)
- const `mapped_type` & `at` (const `key_type` &\_\_k) const
- `const_local_iterator` `begin` (`size_type` \_\_n) const
- `const_local_iterator` `cbegin` (`size_type` \_\_n) const
- `const_local_iterator` `end` (`size_type` \_\_n) const
- `const_local_iterator` `cend` (`size_type` \_\_n) const

## Friends

- template<typename \_Key1, typename \_Tp1, typename \_Hash1, typename \_Pred1, typename \_Alloc1>  
bool **operator==** (const `unordered_map`< \_Key1, \_Tp1, \_Hash1, \_Pred1, \_Alloc1 > &, const `unordered_map`< \_Key1, \_Tp1, \_Hash1, \_Pred1, \_Alloc1 > &)

### 5.998.1 Detailed Description

template<typename \_Key, typename \_Tp, typename \_Hash = hash<\_Key>, typename \_Pred = equal\_to<\_Key>, typename \_Alloc = allocator<std::pair<const \_Key, \_Tp>>>>  
class std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Since

C++11

## Template Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>_Key</code>  | Type of key objects.                                  |
| <code>_Tp</code>   | Type of mapped objects.                               |
| <code>_Hash</code> | Hashing function object type, defaults to hash<_Key>. |

|                     |                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------|
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Key&gt;</code> .                   |
| <code>_Alloc</code> | Allocator type, defaults to <code>std::allocator&lt;std::pair&lt;const _Key, _Tp&gt;&gt;</code> . |

Meets the requirements of a [container](#), and [unordered associative container](#)

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__umap_hashtable`.

## 5.998.2 Member Typedef Documentation

### `allocator_type`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type
Public typedefs.
```

### `const_iterator`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator
Iterator-related typedefs.
```

### `const_local_iterator`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_local_iterator std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator
Iterator-related typedefs.
```

### `const_pointer`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_pointer std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_pointer
Iterator-related typedefs.
```

### `const_reference`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_reference
Iterator-related typedefs.
```

### `difference_type`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::difference_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::difference_type
```

Iterator-related typedefs.

### hasher

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::hasher std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

### iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

### key\_equal

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::key_equal std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

### key\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::key_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

### local\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator
```

Iterator-related typedefs.

### mapped\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::mapped_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::mapped_type
```

Public typedefs.

### pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

**reference**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reference
Iterator-related typedefs.
```

**size\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::size_type std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type
Iterator-related typedefs.
```

**value\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::value_type std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type
Public typedefs.
```

**5.998.3 Constructor & Destructor Documentation****unordered\_map() [1/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map () [default]
Default constructor.
```

**unordered\_map() [2/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    size_type __n,
    const hasher & __hf = hasher(),
    const key_equal & __eqf = key_equal(),
    const allocator_type & __a = allocator_type()) [inline], [explicit]
Default constructor creates no elements.
```

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

**unordered\_map() [3/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
```

```

    __InputIterator __first,
    __InputIterator __last,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() [inline]

```

Builds an unordered\_map from a range.

#### Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eq1</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an unordered\_map consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is `distance(__first, __last)`).

#### unordered\_map() [4/7]

```

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]

```

Copy constructor.

#### unordered\_map() [5/7]

```

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]

```

Move constructor.

#### unordered\_map() [6/7]

```

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    const allocator_type & __a) [inline], [explicit]

```

Creates an unordered\_map with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

#### unordered\_map() [7/7]

```

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>

```



```
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    initializer_list< value_type > __l,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_map from an initializer\_list.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_map consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

### 5.998.4 Member Function Documentation

#### at() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
    const key_type & __k) [inline]
```

Access to unordered\_map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

#### Returns

A reference to the data whose key is equal to `__k`, if such a data is present in the unordered\_map.

#### Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

#### at() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
    const key_type & __k) const [inline]
```

Access to unordered\_map data.

## Parameters

|                        |                                             |
|------------------------|---------------------------------------------|
| <code>_↔<br/>_k</code> | The key for which data should be retrieved. |
|------------------------|---------------------------------------------|

## Returns

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

## Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

**begin()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_map`.

**begin()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `unordered_map`.

**begin()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::begin (
    size_type __n) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

## Parameters

|                        |                   |
|------------------------|-------------------|
| <code>_↔<br/>_n</code> | The bucket index. |
|------------------------|-------------------|

## Returns

A read/write local iterator.

**begin()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::begin (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_map.

**cbegin() [1/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_map.

**cbegin() [2/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**cend() [1/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_map.

**cend()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**clear()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
    const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                 |                                |
|-----------------|--------------------------------|
| <code>_↵</code> | Key of elements to be located. |
| <code>_x</code> |                                |

**Returns**

True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
bool std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
    const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                 |                                |
|-----------------|--------------------------------|
| <code>_↔</code> | Key of elements to be located. |
| <code>_X</code> |                                |

**Returns**

True if there is any element with the specified key.

**count() [1/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
    const _Kt & __x) const -> decltype(_M_h._M_count_tr(__x))    [inline]
```

Finds the number of elements.

**Parameters**

|                 |               |
|-----------------|---------------|
| <code>_↔</code> | Key to count. |
| <code>_X</code> |               |

**Returns**

Number of elements with specified key.

This function only makes sense for `unordered_multimap`; for `unordered_map` the result will either be 0 (not present) or 1 (present).

**count() [2/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
    const key_type & __x) const    [inline]
```

Finds the number of elements.

**Parameters**

|                 |               |
|-----------------|---------------|
| <code>_↔</code> | Key to count. |
| <code>_X</code> |               |

**Returns**

Number of elements with specified key.

This function only makes sense for `unordered_multimap`; for `unordered_map` the result will either be 0 (not present) or 1 (present).

**emplace()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
    _Args &&... __args) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`. Insertion requires amortized constant time.

**emplace\_hint()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                             |
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**empty()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the `unordered_map` is empty.

**end()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_map`.

**end()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the unordered\_map.

**end()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end (
    size_type __n) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read/write local iterator.

**end()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**equal\_range()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Kt>
auto std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::equal_range (
    const _Kt & __x) -> decltype(_M_h._M_equal_range_tr(__x)) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_x</code> |                    |



**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

**equal\_range()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
    const _Kt & __x) const -> decltype(_M_h._M_equal_range_tr(__x))    [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

**equal\_range()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::pair< iterator, iterator > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
    const key_type & __x)    [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

**equal\_range()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::pair< const_iterator, const_iterator > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
    const key_type & __x) const    [inline]
```

Finds a subsequence matching given key.

## Parameters

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered\_multimap.

**erase()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const key_type & __x) [inline]
```

Erases elements according to the provided key.

## Parameters

|                 |                              |
|-----------------|------------------------------|
| <code>_↵</code> | Key of element to be erased. |
| <code>_X</code> |                              |

## Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered\_map. For an unordered\_map the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an unordered\_map.

## Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

## Returns

The iterator `__last`.

This function erases a sequence of elements from an unordered\_map. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const_iterator __position) [inline]
```

Erases an element from an unordered\_map.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    iterator __position) [inline]
```

Erases an element from an unordered\_map.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const _Kt & __x) -> decltype(_M_h._M_find_tr(__x)) [inline]
```

Tries to locate an element in an unordered\_map.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

**find()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Kt>
auto std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x))    [inline]
```

Tries to locate an element in an `unordered_map`.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

**find()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const key_type & __x) [inline]
```

Tries to locate an element in an `unordered_map`.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

**find()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const key_type & __x) const [inline]
```

Tries to locate an element in an `unordered_map`.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↔</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
_allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the `unordered_map`.

**hash\_function()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
_hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the `unordered_map` was constructed.

**insert()** [1/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert()** [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, pair< iterator, bool > > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    _Pair && __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

## Parameters

|                 |                                                                      |
|-----------------|----------------------------------------------------------------------|
| <code>_↔</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
| <code>_X</code> |                                                                      |

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered\_map. An unordered\_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered\_map. Insertion requires amortized constant time.

**insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const value_type & __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

|                 |                                                                      |
|-----------------|----------------------------------------------------------------------|
| <code>_↔</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
| <code>_X</code> |                                                                      |

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered\_map. An unordered\_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered\_map. Insertion requires amortized constant time.

**insert()** [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    _Pair && __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    const value_type & __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

**Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>    | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    value_type && __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

**Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>    | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the `unordered_map`.

**Parameters**

|                  |                                                                                    |
|------------------|------------------------------------------------------------------------------------|
| <code>__l</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
| <code>__l</code> |                                                                                    |
| <code>__l</code> |                                                                                    |
| <code>__l</code> |                                                                                    |
| <code>l</code>   |                                                                                    |

Complexity similar to that of the range constructor.

**insert()** [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, bool > std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    value_type && __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

**Parameters**

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
| <code>x</code>   |                                                                                   |

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a `bool` that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

**key\_eq()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the `unordered_map` was constructed.



**load\_factor()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

**max\_bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const [inline],
[noexcept]
```

Returns the maximum number of buckets of the unordered\_map.

**max\_load\_factor()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_map tries to keep the load factor less than or equal to.

**max\_load\_factor()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
    float __z) [inline]
```

Change the unordered\_map maximum load factor.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__z</code> | The new maximum load factor. |
|------------------|------------------------------|

**max\_size()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_map.

**operator=()** [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]
```

Copy assignment operator.

**operator=()** [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    initializer_list<value_type > __l) [inline]
```

Unordered\_map list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills an unordered\_map with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the unordered\_map and that the resulting unordered\_map's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]
```

Move assignment operator.

**operator[]()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
    const key_type & __k) [inline]
```

Subscript ( [] ) access to unordered\_map data.

**Parameters**

|    |                                             |
|----|---------------------------------------------|
| ↩  | The key for which data should be retrieved. |
| _k |                                             |

**Returns**

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

**operator[]()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
    key_type && __k) [inline]
```

Subscript ( [] ) access to unordered\_map data.

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <code>_↵<br/>_k</code> | The key for which data should be retrieved. |
|------------------------|---------------------------------------------|

**Returns**

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( `[]` ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned. Lookup requires constant time.

**rehash()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵  
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::rehash (   
    size_type __n) [inline]
```

May rehash the unordered\_map.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <code>_↵<br/>_n</code> | The new number of buckets. |
|------------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_map maximum load factor.

**reserve()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵  
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::reserve (   
    size_type __n) [inline]
```

Prepare the unordered\_map for a specified number of elements.

**Parameters**

|                        |                              |
|------------------------|------------------------------|
| <code>_↵<br/>_n</code> | Number of elements required. |
|------------------------|------------------------------|

Same as `rehash(ceil(n / max_load_factor()))`.

**size()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵  
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the unordered\_map.

**swap()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵  
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
```

```
void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::swap (  
    unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> & __x) [inline], [noexcept]
```

Swaps data with another `unordered_map`.

## Parameters

|                 |                                                           |
|-----------------|-----------------------------------------------------------|
| <code>_↔</code> | An unordered_map of the same element and allocator types. |
| <code>_X</code> |                                                           |

This exchanges the elements between two unordered\_map in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 5.999 std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_map>
```

Inheritance diagram for `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`:



### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_multimap >` **const\_iterator**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_multimap >` **const\_local\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator< _Base_iterator, unordered_multimap >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_multimap >` **local\_iterator**
- typedef `_Base::mapped_type` **mapped\_type**
- using **node\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_multimap** (\_Base\_ref \_\_x)
- template<typename \_InputIterator>  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (const allocator\_type &\_\_a)
- **unordered\_multimap** (const unordered\_multimap &)=default
- **unordered\_multimap** (const unordered\_multimap &\_\_umap, const allocator\_type &\_\_a)
- **unordered\_multimap** (initializer\_list<value\_type> \_\_l, const allocator\_type &\_\_a)
- **unordered\_multimap** (initializer\_list<value\_type> \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** (initializer\_list<value\_type> \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multimap** (initializer\_list<value\_type> \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (unordered\_multimap &&)=default
- **unordered\_multimap** (unordered\_multimap &&\_\_umap, const allocator\_type &\_\_a) noexcept(noexcept(\_Base(std::move(\_\_umap), \_\_a)))
- const \_Base & **\_M\_base** () const noexcept
- \_Base & **\_M\_base** () noexcept
- const\_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local\_iterator **begin** (size\_type \_\_b)
- const\_local\_iterator **begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- const\_iterator **cbegin** () const noexcept
- const\_local\_iterator **cbegin** (size\_type \_\_b) const
- const\_iterator **cend** () const noexcept
- const\_local\_iterator **cend** (size\_type \_\_b) const
- void **clear** () noexcept
- template<typename... \_Args>  
iterator **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_hint, \_Args &&... \_\_args)
- const\_iterator **end** () const noexcept
- iterator **end** () noexcept
- local\_iterator **end** (size\_type \_\_b)
- const\_local\_iterator **end** (size\_type \_\_b) const
- template<typename \_Kt, typename = std::\_\_has\_is\_transparent\_t<\_Hash,\_Kt>, typename = std::\_\_has\_is\_transparent\_t<\_Pred,\_Kt>>  
std::pair<iterator, iterator> **equal\_range** (const \_Kt &\_\_k)

- `template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>`  
`std::pair< const_iterator, const_iterator > equal_range (const _Kt &__k) const`
- `std::pair< iterator, iterator > equal_range (const key_type &__key)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__key) const`
- `_Base_iterator erase (_Base_const_iterator __it)`
- `size_type erase (const key_type &__key)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator erase (const_iterator __it)`
- `iterator erase (iterator __it)`
- `node_type extract (const key_type &__key)`
- `node_type extract (const_iterator __position)`
- `template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>`  
`iterator find (const _Kt &__k)`
- `template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>`  
`const_iterator find (const _Kt &__k) const`
- `iterator find (const key_type &__key)`
- `const_iterator find (const key_type &__key) const`
- `template<typename _InputIterator>`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert (_Pair &&__obj)`
- `iterator insert (const value_type &__obj)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert (const_iterator __hint, _Pair &&__obj)`
- `iterator insert (const_iterator __hint, const value_type &__obj)`
- `iterator insert (const_iterator __hint, node_type &&__nh)`
- `iterator insert (const_iterator __hint, value_type &&__x)`
- `iterator insert (node_type &&__nh)`
- `void insert (std::initializer_list< value_type > __l)`
- `iterator insert (value_type &&__x)`
- `float max_load_factor () const noexcept`
- `void max_load_factor (float __f)`
- `template<typename _H2, typename _P2>`  
`void merge (unordered_map< _Key, _Tp, _H2, _P2, _Alloc > &&__source)`
- `template<typename _H2, typename _P2>`  
`void merge (unordered_map< _Key, _Tp, _H2, _P2, _Alloc > &__source)`
- `template<typename _H2, typename _P2>`  
`void merge (unordered_multimap< _Key, _Tp, _H2, _P2, _Alloc > &&__source)`
- `template<typename _H2, typename _P2>`  
`void merge (unordered_multimap< _Key, _Tp, _H2, _P2, _Alloc > &__source)`
- `unordered_multimap & operator= (const unordered_multimap &)=default`
- `unordered_multimap & operator= (initializer_list< value_type > __l)`
- `unordered_multimap & operator= (unordered_multimap &&)=default`
- `void swap (unordered_multimap &__x) noexcept(noexcept(declval< _Base & >().swap(__x)))`

## Protected Member Functions

- `constexpr void _M_swap (const _Safe_container &__x) const noexcept`

## Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug:: Safe\_iterator
- template<typename \_ItT, typename \_SeqT>  
class ::\_\_gnu\_debug:: Safe\_local\_iterator

### 5.999.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal<_Key>,
typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>>
class std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>
```

Class std::unordered\_multimap with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 5.1000 std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_map>
```

## Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::mapped\_type [mapped\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef \_Hashtable::pointer [pointer](#)
- typedef \_Hashtable::const\_pointer [const\\_pointer](#)
- typedef \_Hashtable::reference [reference](#)
- typedef \_Hashtable::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- [unordered\\_multimap](#) ()=default
- template<typename \_InputIterator>  
[unordered\\_multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator>  
[unordered\\_multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator>  
[unordered\\_multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)



- `template<typename _InputIterator>`  
`unordered_multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_multimap` (const `allocator_type` &\_\_a)
- `unordered_multimap` (const `unordered_multimap` &)=default
- `unordered_multimap` (const `unordered_multimap` &\_\_ummap, const `allocator_type` &\_\_a)
- `unordered_multimap` (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a)
- `unordered_multimap` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, const `allocator_type` &\_\_a)
- `unordered_multimap` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `unordered_multimap` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_multimap` (`size_type` \_\_n, const `allocator_type` &\_\_a)
- `unordered_multimap` (`size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `unordered_multimap` (`size_type` \_\_n, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_multimap` (`unordered_multimap` &&)=default
- `unordered_multimap` (`unordered_multimap` &&\_\_ummap, const `allocator_type` &\_\_a) noexcept(noexcept(\_↵ Hashtable(`std::move`(\_\_ummap.\_M\_h), \_\_a)))
- `iterator` `begin` () noexcept
- `local_iterator` `begin` (`size_type` \_\_n)
- `size_type` `bucket` (const `key_type` &\_\_key) const
- `size_type` `bucket_count` () const noexcept
- `size_type` `bucket_size` (`size_type` \_\_n) const
- void `clear` () noexcept
- `template<typename... _Args>`  
`iterator` `emplace` (`_Args` &&... \_\_args)
- `template<typename... _Args>`  
`iterator` `emplace_hint` (const `iterator` \_\_pos, `_Args` &&... \_\_args)
- bool `empty` () const noexcept
- `iterator` `end` () noexcept
- `local_iterator` `end` (`size_type` \_\_n)
- `size_type` `erase` (const `key_type` &\_\_x)
- `iterator` `erase` (const `iterator` \_\_first, const `iterator` \_\_last)
- `allocator_type` `get_allocator` () const noexcept
- `hasher` `hash_function` () const
- `template<typename _InputIterator>`  
void `insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void `insert` (`initializer_list`< `value_type` > \_\_l)
- `key_equal` `key_eq` () const
- float `load_factor` () const noexcept
- `size_type` `max_bucket_count` () const noexcept
- float `max_load_factor` () const noexcept
- void `max_load_factor` (float \_\_z)
- `size_type` `max_size` () const noexcept
- `unordered_multimap` & `operator=` (const `unordered_multimap` &)=default
- `unordered_multimap` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- `unordered_multimap` & `operator=` (`unordered_multimap` &&)=default
- void `rehash` (`size_type` \_\_n)
- void `reserve` (`size_type` \_\_n)
- `size_type` `size` () const noexcept

- void `swap` (`unordered_multimap` &\_\_x) noexcept(noexcept(\_M\_h.swap(\_\_x.\_M\_h)))
- `const_iterator begin` () const noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator end` () const noexcept
- `const_iterator cend` () const noexcept
- `iterator insert` (const `value_type` &\_\_x)
- `iterator insert` (`value_type` &&\_\_x)
- template<typename \_Pair>  
\_\_enable\_if\_t< `is_constructible`< `value_type`, \_Pair && >::value, `iterator` > `insert` (\_Pair &&\_\_x)
- `iterator insert` (`const_iterator` \_\_hint, const `value_type` &\_\_x)
- `iterator insert` (`const_iterator` \_\_hint, `value_type` &&\_\_x)
- template<typename \_Pair>  
\_\_enable\_if\_t< `is_constructible`< `value_type`, \_Pair && >::value, `iterator` > `insert` (`const_iterator` \_\_hint, \_Pair &&\_\_x)
- `iterator erase` (`const_iterator` \_\_position)
- `iterator erase` (`iterator` \_\_position)
- `iterator find` (const `key_type` &\_\_x)
- template<typename \_Kt>  
auto `find` (const \_Kt &\_\_x) -> decltype(\_M\_h.\_M\_find\_tr(\_\_x))
- `const_iterator find` (const `key_type` &\_\_x) const
- template<typename \_Kt>  
auto `find` (const \_Kt &\_\_x) const -> decltype(\_M\_h.\_M\_find\_tr(\_\_x))
- `size_type count` (const `key_type` &\_\_x) const
- template<typename \_Kt>  
auto `count` (const \_Kt &\_\_x) const -> decltype(\_M\_h.\_M\_count\_tr(\_\_x))
- bool `contains` (const `key_type` &\_\_x) const
- template<typename \_Kt>  
auto `contains` (const \_Kt &\_\_x) const -> decltype(\_M\_h.\_M\_find\_tr(\_\_x), void(), true)
- `std::pair`< `iterator`, `iterator` > `equal_range` (const `key_type` &\_\_x)
- template<typename \_Kt>  
auto `equal_range` (const \_Kt &\_\_x) -> decltype(\_M\_h.\_M\_equal\_range\_tr(\_\_x))

- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const
- `template<typename _Kt>`  
`auto equal_range` (const \_Kt &\_\_x) const -> `decltype(_M_h._M_equal_range_tr(__x))`
- `const_local_iterator begin` (size\_type \_\_n) const
- `const_local_iterator cbegin` (size\_type \_\_n) const
- `const_local_iterator end` (size\_type \_\_n) const
- `const_local_iterator cend` (size\_type \_\_n) const

## Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1>`  
`bool operator==` (const `unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &)

## 5.1000.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>`  
`class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Since

C++11

## Template Parameters

|                     |                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------|
| <code>_Key</code>   | Type of key objects.                                                                              |
| <code>_Tp</code>    | Type of mapped objects.                                                                           |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Key&gt;</code> .                         |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Key&gt;</code> .                   |
| <code>_Alloc</code> | Allocator type, defaults to <code>std::allocator&lt;std::pair&lt;const _Key, _Tp&gt;&gt;</code> . |

Meets the requirements of a `container`, and `unordered associative container`

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__ummap_hashtable`.

## 5.1000.2 Member Typedef Documentation

### `allocator_type`

`template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>`  
`typedef _Hashtable::allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`  
`::allocator_type`

Public typedefs.

**const\_iterator**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator
```

Iterator-related typedefs.

**const\_local\_iterator**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator
```

Iterator-related typedefs.

**const\_pointer**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_pointer
```

Iterator-related typedefs.

**const\_reference**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_reference
```

Iterator-related typedefs.

**difference\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::difference_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::difference_type
```

Iterator-related typedefs.

**hasher**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

**iterator**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

**key\_equal**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
Public typedefs.
```

**key\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
Public typedefs.
```

**local\_iterator**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator
Iterator-related typedefs.
```

**mapped\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::mapped_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::mapped_type
Public typedefs.
```

**pointer**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer
Iterator-related typedefs.
```

**reference**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reference
Iterator-related typedefs.
```

**size\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type
Iterator-related typedefs.
```

**value\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::value_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::value_type
Public typedefs.
```

**5.1000.3 Constructor & Destructor Documentation****unordered\_multimap() [1/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap () [default]
Default constructor.
```

**unordered\_multimap() [2/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    size_type __n,
    const hasher & __hf = hasher(),
    const key_equal & __eqf = key_equal(),
    const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Default constructor creates no elements.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

**unordered\_multimap() [3/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    _InputIterator __first,
    _InputIterator __last,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eqf = key_equal(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_multimap from a range.

**Parameters**

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |

## Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__hf</code>  | A hash functor.         |
| <code>__eqf</code> | A key equality functor. |
| <code>__a</code>   | An allocator object.    |

Create an `unordered_multimap` consisting of copies of the elements from `[__first,__last)`. This is linear in N (where N is `distance(__first,__last)`).

**unordered\_multimap()** [4/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

**unordered\_multimap()** [5/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

**unordered\_multimap()** [6/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    const allocator_type & __a ) [inline], [explicit]
```

Creates an `unordered_multimap` with no elements.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

**unordered\_multimap()** [7/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    initializer_list< value_type > __l,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eqf = key_equal(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_multimap` from an `initializer_list`.

## Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> . |
|------------------|------------------------------------|

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an `unordered_multimap` consisting of copies of the elements in the list. This is linear in  $N$  (where  $N$  is `__l.size()`).

## 5.1000.4 Member Function Documentation

**begin()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_multimap`.

**begin()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `unordered_multimap`.

**begin()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::begin (
    size_type __n) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

## Returns

A read/write local iterator.

**begin()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::begin (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.



**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_multimap.

**cbegin() [1/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

**cbegin() [2/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**cend() [1/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multimap.

**cend()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

**clear()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
    const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

**Returns**

True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
    const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                 |                                |
|-----------------|--------------------------------|
| <code>_↵</code> | Key of elements to be located. |
| <code>_X</code> |                                |

**Returns**

True if there is any element with the specified key.

**count()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
    const _Kt & __x) const -> decltype(_M_h._M_count_tr(__x))    [inline]
```

Finds the number of elements.

**Parameters**

|                 |               |
|-----------------|---------------|
| <code>_↵</code> | Key to count. |
| <code>_X</code> |               |

**Returns**

Number of elements with specified key.

**count()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
    const key_type & __x) const    [inline]
```

Finds the number of elements.

**Parameters**

|                 |               |
|-----------------|---------------|
| <code>_↵</code> | Key to count. |
| <code>_X</code> |               |

**Returns**

Number of elements with specified key.

**emplace()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
    _Args &&... __args)    [inline]
```

Attempts to build and insert a std::pair into the unordered\_multimap.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the `unordered_multimap`. Insertion requires amortized constant time.

**emplace\_hint()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_multimap`.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                             |
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the element with key of the `std::pair` built from `__args`.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**empty()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the `unordered_multimap` is empty.

**end() [1/4]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multimap`.

**end()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the unordered\_multimap.

**end()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
    size_type __n) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read/write local iterator.

**end()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**equal\_range()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
    const _Kt & __x) -> decltype(_M_h._M_equal_range_tr(__x)) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_x</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**equal\_range()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
    const _Kt & __x) const -> decltype(_M_h._M_equal_range_tr(__x))    [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**equal\_range()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, iterator > std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::
equal_range (
    const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**equal\_range()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< const_iterator, const_iterator > std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::
equal_range (
    const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**erase()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                               |
|------------------|-------------------------------|
| <code>__x</code> | Key of elements to be erased. |
|------------------|-------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multimap`.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multimap`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const_iterator __position) [inline]
```

Erases an element from an `unordered_multimap`.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    iterator __position) [inline]
```

Erases an element from an `unordered_multimap`.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const _Kt & __x) -> decltype(_M_h._M_find_tr(__x)) [inline]
```

Tries to locate an element in an `unordered_multimap`.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.



**find()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x))    [inline]
```

Tries to locate an element in an unordered\_multimap.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const key_type & __x) [inline]
```

Tries to locate an element in an unordered\_multimap.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_multimap.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered\_multimap.

**hash\_function()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the unordered\_multimap was constructed.

**insert()** [1/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert()** [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    _Pair && __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

**insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const value_type & __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

**insert()** [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    _Pair && __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the element with key of \_\_x (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
```

```
    const_iterator __hint,  
    const value_type & __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>    | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    value_type && __x) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

**Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>    | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the `unordered_multimap`.

**Parameters**

|                  |                                                                                    |
|------------------|------------------------------------------------------------------------------------|
| <code>__l</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
|------------------|------------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

**insert()** [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    value_type && __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

**key\_eq()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the unordered\_multimap was constructed.

**load\_factor()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

**max\_bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered\_multimap.

**max\_load\_factor()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_multimap tries to keep the load factor less than or equal to.

**max\_load\_factor()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
    float __z) [inline]
```

Change the unordered\_multimap maximum load factor.

## Parameters

|                 |                              |
|-----------------|------------------------------|
| <code>_↵</code> | The new maximum load factor. |
| <code>_Z</code> |                              |

**max\_size()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_multimap.

**operator=()** [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]
```

Copy assignment operator.

**operator=()** [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    initializer_list< value_type > __l) [inline]
```

Unordered\_multimap list assignment operator.

## Parameters

|                                   |                      |
|-----------------------------------|----------------------|
| <code>↵</code>                    | An initializer_list. |
| <code>_↵</code>                   |                      |
| <code>↵</code>                    |                      |
| <code>_↵</code><br><code>/</code> |                      |

This function fills an unordered\_multimap with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_multimap and that the resulting unordered\_multimap's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]
```

Move assignment operator.

**rehash()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```



```
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
    size_type __n) [inline]
```

May rehash the unordered\_multimap.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__n</code> | The new number of buckets. |
|------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_multimap maximum load factor.

#### reserve()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
    size_type __n) [inline]
```

Prepare the unordered\_multimap for a specified number of elements.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

Same as rehash(ceil(n / max\_load\_factor())).

#### size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size () const [inline],
[noexcept]
```

Returns the size of the unordered\_multimap.

#### swap()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
    unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another unordered\_multimap.

#### Parameters

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>__x</code> | An unordered_multimap of the same element and allocator types. |
|------------------|----------------------------------------------------------------|

This exchanges the elements between two unordered\_multimap in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 5.1001 `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

```
#include <unordered_set>
```

Inheritance diagram for `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`:



### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_multiset >` **const\_iterator**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_multiset >` **const\_local\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator< _Base_iterator, unordered_multiset >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_multiset >` **local\_iterator**
- using **node\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

### Public Member Functions

- **unordered\_multiset** (`_Base_ref __x`)
- template<typename `_InputIterator`>  
**unordered\_multiset** (`_InputIterator __first, _InputIterator __last, const allocator_type &__a`)
- template<typename `_InputIterator`>  
**unordered\_multiset** (`_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a`)
- template<typename `_InputIterator`>  
**unordered\_multiset** (`_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a`)
- template<typename `_InputIterator`>  
**unordered\_multiset** (`_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- **unordered\_multiset** (`const allocator_type &__a`)
- **unordered\_multiset** (`const unordered_multiset &`)=default

- **unordered\_multiset** (const [unordered\\_multiset](#) &\_\_uset, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** ([unordered\\_multiset](#) &&)=default
- **unordered\_multiset** ([unordered\\_multiset](#) && \_\_uset, const allocator\_type &\_\_a) noexcept(noexcept([\\_Base](#)(std::move(↵\_\_uset), \_\_a)))
- const [\\_Base](#) & **\_M\_base** () const noexcept
- [\\_Base](#) & **\_M\_base** () noexcept
- [const\\_iterator](#) **begin** () const noexcept
- [iterator](#) **begin** () noexcept
- [local\\_iterator](#) **begin** (size\_type \_\_b)
- [const\\_local\\_iterator](#) **begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- [const\\_iterator](#) **cbegin** () const noexcept
- [const\\_local\\_iterator](#) **cbegin** (size\_type \_\_b) const
- [const\\_iterator](#) **cend** () const noexcept
- [const\\_local\\_iterator](#) **cend** (size\_type \_\_b) const
- void **clear** () noexcept
- template<typename... \_Args>  
[iterator](#) **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator](#) **emplace\_hint** (const [iterator](#) \_\_hint, \_Args &&... \_\_args)
- [const\\_iterator](#) **end** () const noexcept
- [iterator](#) **end** () noexcept
- [local\\_iterator](#) **end** (size\_type \_\_b)
- [const\\_local\\_iterator](#) **end** (size\_type \_\_b) const
- template<typename \_Kt, typename = std::\_\_has\_is\_transparent\_t<\_Hash, \_Kt>, typename = std::\_\_has\_is\_transparent\_t<\_Pred, \_Kt>>  
[std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const \_Kt &\_\_k)
- template<typename \_Kt, typename = std::\_\_has\_is\_transparent\_t<\_Hash, \_Kt>, typename = std::\_\_has\_is\_transparent\_t<\_Pred, \_Kt>>  
[std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > **equal\_range** (const \_Kt &\_\_k) const
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_key)
- [std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > **equal\_range** (const key\_type &\_\_key) const
- [\\_Base\\_iterator](#) **erase** ([\\_Base\\_const\\_iterator](#) \_\_it)
- size\_type **erase** (const key\_type &\_\_key)
- [iterator](#) **erase** ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
- [iterator](#) **erase** ([const\\_iterator](#) \_\_it)
- [iterator](#) **erase** ([iterator](#) \_\_it)
- node\_type **extract** (const key\_type &\_\_key)
- node\_type **extract** ([const\\_iterator](#) \_\_position)
- template<typename \_Kt, typename = std::\_\_has\_is\_transparent\_t<\_Hash, \_Kt>, typename = std::\_\_has\_is\_transparent\_t<\_Pred, \_Kt>>  
[iterator](#) **find** (const \_Kt &\_\_k)
- template<typename \_Kt, typename = std::\_\_has\_is\_transparent\_t<\_Hash, \_Kt>, typename = std::\_\_has\_is\_transparent\_t<\_Pred, \_Kt>>  
[const\\_iterator](#) **find** (const \_Kt &\_\_k) const

- [iterator find](#) (const key\_type &\_\_key)
- [const\\_iterator find](#) (const key\_type &\_\_key) const
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator insert](#) (const value\_type &\_\_obj)
- [iterator insert](#) ([const\\_iterator](#) \_\_hint, const value\_type &\_\_obj)
- [iterator insert](#) ([const\\_iterator](#) \_\_hint, node\_type &&\_\_nh)
- [iterator insert](#) ([const\\_iterator](#) \_\_hint, value\_type &&\_\_obj)
- [iterator insert](#) (node\_type &&\_\_nh)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [iterator insert](#) (value\_type &&\_\_obj)
- float **max\_load\_factor** () const noexcept
- void **max\_load\_factor** (float \_\_f)
- template<typename \_H2, typename \_P2>  
void **merge** ([unordered\\_multiset](#)< \_Value, \_H2, \_P2, \_Alloc > &&\_\_source)
- template<typename \_H2, typename \_P2>  
void **merge** ([unordered\\_multiset](#)< \_Value, \_H2, \_P2, \_Alloc > &\_\_source)
- template<typename \_H2, typename \_P2>  
void **merge** ([unordered\\_set](#)< \_Value, \_H2, \_P2, \_Alloc > &&\_\_source)
- template<typename \_H2, typename \_P2>  
void **merge** ([unordered\\_set](#)< \_Value, \_H2, \_P2, \_Alloc > &\_\_source)
- [unordered\\_multiset](#) & **operator=** (const [unordered\\_multiset](#) &)=default
- [unordered\\_multiset](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_multiset](#) & **operator=** ([unordered\\_multiset](#) &&)=default
- void **swap** ([unordered\\_multiset](#) &\_\_x) noexcept(noexcept([declval](#)< \_Base & >().swap(\_\_x)))

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug::Safe\_iterator
- template<typename \_ItT, typename \_SeqT>  
class ::\_\_gnu\_debug::Safe\_local\_iterator

#### 5.1001.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>,
typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

Class std::unordered\_multiset with safety/checking/debug instrumentation.  
The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 5.1002 std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_set>
```

## Public Types

- typedef `_Hashtable::key_type` [key\\_type](#)
- typedef `_Hashtable::value_type` [value\\_type](#)
- typedef `_Hashtable::hasher` [hasher](#)
- typedef `_Hashtable::key_equal` [key\\_equal](#)
- typedef `_Hashtable::allocator_type` [allocator\\_type](#)
- typedef `_Hashtable::pointer` [pointer](#)
- typedef `_Hashtable::const_pointer` [const\\_pointer](#)
- typedef `_Hashtable::reference` [reference](#)
- typedef `_Hashtable::const_reference` [const\\_reference](#)
- typedef `_Hashtable::iterator` [iterator](#)
- typedef `_Hashtable::const_iterator` [const\\_iterator](#)
- typedef `_Hashtable::local_iterator` [local\\_iterator](#)
- typedef `_Hashtable::const_local_iterator` [const\\_local\\_iterator](#)
- typedef `_Hashtable::size_type` [size\\_type](#)
- typedef `_Hashtable::difference_type` [difference\\_type](#)

## Public Member Functions

- [unordered\\_multiset](#) ()=default
- template<typename `_InputIterator`>  
**unordered\_multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const [allocator\\_type](#) &\_\_a)
- template<typename `_InputIterator`>  
**unordered\_multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- template<typename `_InputIterator`>  
**unordered\_multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- template<typename `_InputIterator`>  
**unordered\_multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multiset](#) (const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) (const [unordered\\_multiset](#) &)=default
- **unordered\_multiset** (const [unordered\\_multiset](#) &\_\_umset, const [allocator\\_type](#) &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **unordered\_multiset** ([size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- **unordered\_multiset** ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- **unordered\_multiset** ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multiset](#) ([unordered\\_multiset](#) &&)=default
- **unordered\_multiset** ([unordered\\_multiset](#) &&\_\_umset, const [allocator\\_type](#) &\_\_a) noexcept(noexcept(`std::move`(\_\_umset.\_M\_h), \_\_a))
- [size\\_type](#) **bucket** (const [key\\_type](#) &\_\_key) const
- [size\\_type](#) **bucket\_count** () const noexcept

- [size\\_type bucket\\_size](#) ([size\\_type](#) \_\_n) const
  - [const\\_iterator cbegin](#) () const noexcept
  - [const\\_iterator cend](#) () const noexcept
  - void [clear](#) () noexcept
  - template<typename... \_Args>  
[iterator emplace](#) (\_Args &&... \_\_args)
  - template<typename... \_Args>  
[iterator emplace\\_hint](#) ([const\\_iterator](#) \_\_pos, \_Args &&... \_\_args)
  - bool [empty](#) () const noexcept
  - [size\\_type erase](#) (const [key\\_type](#) &\_\_x)
  - [iterator erase](#) ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
  - [allocator\\_type get\\_allocator](#) () const noexcept
  - [hasher hash\\_function](#) () const
  - template<typename \_InputIterator>  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void [insert](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - [key\\_equal key\\_eq](#) () const
  - float [load\\_factor](#) () const noexcept
  - [size\\_type max\\_bucket\\_count](#) () const noexcept
  - float [max\\_load\\_factor](#) () const noexcept
  - void [max\\_load\\_factor](#) (float \_\_z)
  - [size\\_type max\\_size](#) () const noexcept
  - [unordered\\_multiset](#) & [operator=](#) (const [unordered\\_multiset](#) &)=default
  - [unordered\\_multiset](#) & [operator=](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - [unordered\\_multiset](#) & [operator=](#) ([unordered\\_multiset](#) &&)=default
  - void [rehash](#) ([size\\_type](#) \_\_n)
  - void [reserve](#) ([size\\_type](#) \_\_n)
  - [size\\_type size](#) () const noexcept
  - void [swap](#) ([unordered\\_multiset](#) &\_\_x) noexcept(noexcept(\_\_M\_h.swap(\_\_x.\_M\_h)))
- 
- [iterator begin](#) () noexcept
  - [const\\_iterator begin](#) () const noexcept
- 
- [iterator end](#) () noexcept
  - [const\\_iterator end](#) () const noexcept
- 
- [iterator insert](#) (const [value\\_type](#) &\_\_x)
  - [iterator insert](#) ([value\\_type](#) &&\_\_x)
- 
- [iterator insert](#) ([const\\_iterator](#) \_\_hint, const [value\\_type](#) &\_\_x)
  - [iterator insert](#) ([const\\_iterator](#) \_\_hint, [value\\_type](#) &&\_\_x)
- 
- [iterator erase](#) ([const\\_iterator](#) \_\_position)
  - [iterator erase](#) ([iterator](#) \_\_position)

- `iterator find` (const `key_type` &\_\_x)
- `template<typename _Kt>`  
`auto find` (const `_Kt` &\_\_x) -> `decltype(_M_h._M_find_tr(__x))`
- `const_iterator find` (const `key_type` &\_\_x) const
- `template<typename _Kt>`  
`auto find` (const `_Kt` &\_\_x) const -> `decltype(_M_h._M_find_tr(__x))`
- `size_type count` (const `key_type` &\_\_x) const
- `template<typename _Kt>`  
`auto count` (const `_Kt` &\_\_x) const -> `decltype(_M_h._M_count_tr(__x))`
- `bool contains` (const `key_type` &\_\_x) const
- `template<typename _Kt>`  
`auto contains` (const `_Kt` &\_\_x) const -> `decltype(_M_h._M_find_tr(__x), void(), true)`
- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
- `template<typename _Kt>`  
`auto equal_range` (const `_Kt` &\_\_x) -> `decltype(_M_h._M_equal_range_tr(__x))`
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const
- `template<typename _Kt>`  
`auto equal_range` (const `_Kt` &\_\_x) const -> `decltype(_M_h._M_equal_range_tr(__x))`
- `local_iterator begin` (`size_type` \_\_n)
- `const_local_iterator begin` (`size_type` \_\_n) const
- `const_local_iterator cbegin` (`size_type` \_\_n) const
- `local_iterator end` (`size_type` \_\_n)
- `const_local_iterator end` (`size_type` \_\_n) const
- `const_local_iterator cend` (`size_type` \_\_n) const

## Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1>`  
`bool operator==` (const `unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 >` &)

### 5.1002.1 Detailed Description

`template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc = allocator<_Value>>`  
`class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Since

C++11

## Template Parameters

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>_Value</code> | Type of key objects.                                                              |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> . |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .                  |

Meets the requirements of a [container](#), and [unordered associative container](#)  
 Base is `_Hashtable`, dispatched at compile time via template alias `__umset_hashtable`.

## 5.1002.2 Member Typedef Documentation

**allocator\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >↔
::allocator_type
Public typedefs.
```

**const\_iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >↔
::const_iterator
Iterator-related typedefs.
```

**const\_local\_iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc
>::const_local_iterator
Iterator-related typedefs.
```

**const\_pointer**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >::const↔
_pointer
Iterator-related typedefs.
```

**const\_reference**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >↔
::const_reference
Iterator-related typedefs.
```



### difference\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵
::difference_type
Iterator-related typedefs.
```

### hasher

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hasher
Public typedefs.
```

### iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::iterator
Iterator-related typedefs.
```

### key\_equal

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_equal
Public typedefs.
```

### key\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_type
Public typedefs.
```

### local\_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵
::local_iterator
Iterator-related typedefs.
```

### pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::pointer
Iterator-related typedefs.
```

### reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

### size\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

### value\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::value_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

## 5.1002.3 Constructor & Destructor Documentation

### unordered\_multiset() [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset () [default]
```

Default constructor.

### unordered\_multiset() [2/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    size_type __n,
    const hasher & __hf = hasher(),
    const key_equal & __eqf = key_equal(),
    const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Default constructor creates no elements.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

### unordered\_multiset() [3/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    _InputIterator __first,
    _InputIterator __last,
    size_type __n = 0,
    const hasher & __hf = hasher(),
```

```
const key_equal & __eq1 = key_equal(),
const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_multiset` from a range.

#### Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eq1</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an `unordered_multiset` consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  (where  $N$  is `distance(__first,__last)`).

#### `unordered_multiset()` [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

#### `unordered_multiset()` [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    unordered_multiset< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

#### `unordered_multiset()` [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    initializer_list< value_type > __l,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_multiset` from an `initializer_list`.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An <code>initializer_list</code> . |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an `unordered_multiset` consisting of copies of the elements in the list. This is linear in  $N$  (where  $N$  is `__l.size()`).

**unordered\_multiset()** [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    const allocator_type & __a) [inline], [explicit]
```

Creates an unordered\_multiset with no elements.

**Parameters**

|                                                                                                    |                      |
|----------------------------------------------------------------------------------------------------|----------------------|
|  <code>__a</code> | An allocator object. |
|----------------------------------------------------------------------------------------------------|----------------------|

**5.1002.4 Member Function Documentation****begin()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

**begin()** [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

**begin()** [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
    size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                                                                                                      |                   |
|------------------------------------------------------------------------------------------------------|-------------------|
|  <code>__n</code> | The bucket index. |
|------------------------------------------------------------------------------------------------------|-------------------|

**Returns**

A read-only local iterator.

**begin()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                      |                   |
|----------------------|-------------------|
| $\_ \leftrightarrow$ | The bucket index. |
| $\_n$                |                   |

**Returns**

A read-only local iterator.

**bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_multiset.

**cbegin() [1/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

**cbegin() [2/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin (
size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                      |                   |
|----------------------|-------------------|
| $\_ \leftrightarrow$ | The bucket index. |
| $\_n$                |                   |

**Returns**

A read-only local iterator.

**cend() [1/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

**cend()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

**clear()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_multiset.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::contains (
    const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

**Returns**

True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::contains (
    const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                 |                                |
|-----------------|--------------------------------|
| <code>_↔</code> | Key of elements to be located. |
| <code>_X</code> |                                |

**Returns**

True if there is any element with the specified key.

**count()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (
    const _Kt & __x) const -> decltype(_M_h._M_count_tr(__x))    [inline]
```

Finds the number of elements.

**Parameters**

|                 |                     |
|-----------------|---------------------|
| <code>_↔</code> | Element to located. |
| <code>_X</code> |                     |

**Returns**

Number of elements with specified key.

**count()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (
    const key_type & __x) const    [inline]
```

Finds the number of elements.

**Parameters**

|                 |                     |
|-----------------|---------------------|
| <code>_↔</code> | Element to located. |
| <code>_X</code> |                     |

**Returns**

Number of elements with specified key.

**emplace()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace (
    _Args &&... __args)    [inline]
```

Builds and insert an element into the unordered\_multiset.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

**emplace\_hint()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

**Returns**

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant time.

**empty()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered\_multiset is empty.

**end() [1/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

**end() [2/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.



**end()** [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
    size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|       |                   |
|-------|-------------------|
| $\_n$ | The bucket index. |
|-------|-------------------|

**Returns**

A read-only local iterator.

**end()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|       |                   |
|-------|-------------------|
| $\_n$ | The bucket index. |
|-------|-------------------|

**Returns**

A read-only local iterator.

**equal\_range()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal_range (
    const _Kt & __x) -> decltype(_M_h._M_equal_range_tr(__x)) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|       |                    |
|-------|--------------------|
| $\_x$ | Key to be located. |
|-------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**equal\_range()** [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal_range (
    const _Kt & __x) const -> decltype(_M_h._M_equal_range_tr(__x))    [inline]
```

Finds a subsequence matching given key.

**Parameters**

|       |                    |
|-------|--------------------|
| $\_x$ | Key to be located. |
|-------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**equal\_range()** [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, iterator > std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal←
_range (
    const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|       |                    |
|-------|--------------------|
| $\_x$ | Key to be located. |
|-------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**equal\_range()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< const_iterator, const_iterator > std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::equal_range (
    const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|       |                    |
|-------|--------------------|
| $\_x$ | Key to be located. |
|-------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**erase()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
    const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multiset`.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
    const_iterator __position) [inline]
```

Erases an element from an `unordered_multiset`.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
    iterator __position) [inline]
```

Erases an element from an `unordered_multiset`.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
    const _Kt & __x) -> decltype(_M.h._M_find_tr(__x)) [inline]
```

Tries to locate an element in an `unordered_multiset`.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**find()** [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
```

```
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (  
    const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x))    [inline]
```

Tries to locate an element in an unordered\_multiset.

**Parameters**

|       |                        |
|-------|------------------------|
| $\_↔$ | Element to be located. |
| $\_X$ |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find() [3/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
    const key_type & __x) [inline]
```

Tries to locate an element in an unordered\_multiset.

**Parameters**

|       |                        |
|-------|------------------------|
| $\_↔$ | Element to be located. |
| $\_X$ |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find() [4/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
    const key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_multiset.

**Parameters**

|       |                        |
|-------|------------------------|
| $\_↔$ | Element to be located. |
| $\_X$ |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**get\_allocator()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered\_multiset.

**hash\_function()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the unordered\_multiset was constructed.

**insert() [1/6]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

A template function that inserts a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert() [2/6]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    const value_type & __x) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

**insert()** [3/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    const value_type & __x) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

**insert()** [4/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    value_type && __x) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

**insert()** [5/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    initializer_list< value_type > __l) [inline]
```

Inserts a list of elements into the unordered\_multiset.



**Parameters**

|   |                                                                                    |
|---|------------------------------------------------------------------------------------|
| ↩ | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
| ↩ |                                                                                    |
| ↩ |                                                                                    |
| ↩ |                                                                                    |
| / |                                                                                    |

Complexity similar to that of the range constructor.

**insert()** [6/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    value_type && __x) [inline]
```

Inserts an element into the `unordered_multiset`.

**Parameters**

|     |                         |
|-----|-------------------------|
| ↩   | Element to be inserted. |
| __x |                         |

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

**key\_eq()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the `unordered_multiset` was constructed.

**load\_factor()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::load_factor () const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

**max\_bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the `unordered_multiset`.

**max\_load\_factor()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_multiset tries to keep the load factor less than or equal to.

**max\_load\_factor()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
    float __z) [inline]
```

Change the unordered\_multiset maximum load factor.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__z</code> | The new maximum load factor. |
|------------------|------------------------------|

**max\_size()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_size () const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_multiset.

**operator=()** [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
    const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy assignment operator.

**operator=()** [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
    initializer_list< value_type > __l) [inline]
```

Unordered\_multiset list assignment operator.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills an unordered\_multiset with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the `unordered_multiset` and that the resulting `unordered_multiset`'s size is the same as the number of elements assigned.

### **operator=()** [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
    unordered_multiset< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move assignment operator.

### **rehash()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::rehash (
    size_type __n) [inline]
```

May rehash the `unordered_multiset`.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__n</code> | The new number of buckets. |
|------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the `unordered_multiset` maximum load factor.

### **reserve()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve (
    size_type __n) [inline]
```

Prepare the `unordered_multiset` for a specified number of elements.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

Same as `rehash(ceil(n / max_load_factor()))`.

### **size()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the `unordered_multiset`.

### **swap()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::swap (
    unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another `unordered_multiset`.

## Parameters

|                 |                                                                |
|-----------------|----------------------------------------------------------------|
| <code>_↔</code> | An unordered_multiset of the same element and allocator types. |
| <code>_X</code> |                                                                |

This exchanges the elements between two sets in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 5.1003 std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_set>
```

Inheritance diagram for `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`:



## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_set >` **const\_iterator**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_set >` **const\_local\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- using **insert\_return\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_iterator, unordered_set >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_set >` **local\_iterator**
- using **node\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_set** ([\\_Base\\_ref](#) \_\_x)
- [template](#)<[typename](#) [\\_InputIterator](#)>  
**unordered\_set** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [const allocator\\_type](#) &\_\_a)
- [template](#)<[typename](#) [\\_InputIterator](#)>  
**unordered\_set** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [size\\_type](#) \_\_n, [const allocator\\_type](#) &\_\_a)
- [template](#)<[typename](#) [\\_InputIterator](#)>  
**unordered\_set** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [size\\_type](#) \_\_n, [const hasher](#) &\_\_hf, [const allocator\\_type](#) &\_\_a)
- [template](#)<[typename](#) [\\_InputIterator](#)>  
**unordered\_set** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [size\\_type](#) \_\_n=0, [const hasher](#) &\_\_hf=[hasher](#)(), [const key\\_equal](#) &\_\_eq=[key\\_equal](#)(), [const allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **unordered\_set** ([const allocator\\_type](#) &\_\_a)
- **unordered\_set** ([const unordered\\_set](#) &)=default
- **unordered\_set** ([const unordered\\_set](#) &\_\_uset, [const allocator\\_type](#) &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [const allocator\\_type](#) &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, [const allocator\\_type](#) &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, [const hasher](#) &\_\_hf, [const allocator\\_type](#) &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, [const hasher](#) &\_\_hf=[hasher](#)(), [const key\\_equal](#) &\_\_eq=[key\\_equal](#)(), [const allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **unordered\_set** ([size\\_type](#) \_\_n, [const allocator\\_type](#) &\_\_a)
- **unordered\_set** ([size\\_type](#) \_\_n, [const hasher](#) &\_\_hf, [const allocator\\_type](#) &\_\_a)
- **unordered\_set** ([size\\_type](#) \_\_n, [const hasher](#) &\_\_hf=[hasher](#)(), [const key\\_equal](#) &\_\_eq=[key\\_equal](#)(), [const allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **unordered\_set** ([unordered\\_set](#) &&)=default
- **unordered\_set** ([unordered\\_set](#) &&\_\_uset, [const allocator\\_type](#) &\_\_a) noexcept([noexcept](#)([\\_Base](#)([std::move](#)(\_\_uset), \_\_a)))
- [const](#) [\\_Base](#) & [\\_M\\_base](#) () [const](#) noexcept
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- [const\\_iterator](#) [begin](#) () [const](#) noexcept
- [iterator](#) [begin](#) () noexcept
- [local\\_iterator](#) [begin](#) ([size\\_type](#) \_\_b)
- [const\\_local\\_iterator](#) [begin](#) ([size\\_type](#) \_\_b) [const](#)
- [size\\_type](#) [bucket\\_size](#) ([size\\_type](#) \_\_b) [const](#)
- [const\\_iterator](#) [cbegin](#) () [const](#) noexcept
- [const\\_local\\_iterator](#) [cbegin](#) ([size\\_type](#) \_\_b) [const](#)
- [const\\_iterator](#) [cend](#) () [const](#) noexcept
- [const\\_local\\_iterator](#) [cend](#) ([size\\_type](#) \_\_b) [const](#)
- [void](#) [clear](#) () noexcept
- [template](#)<[typename](#)... [\\_Args](#)>  
[std::pair](#)< [iterator](#), [bool](#) > [emplace](#) ([\\_Args](#) &&... \_\_args)
- [template](#)<[typename](#)... [\\_Args](#)>  
[iterator](#) [emplace\\_hint](#) ([const\\_iterator](#) \_\_hint, [\\_Args](#) &&... \_\_args)
- [const\\_iterator](#) [end](#) () [const](#) noexcept
- [iterator](#) [end](#) () noexcept
- [local\\_iterator](#) [end](#) ([size\\_type](#) \_\_b)
- [const\\_local\\_iterator](#) [end](#) ([size\\_type](#) \_\_b) [const](#)
- [template](#)<[typename](#) [\\_Kt](#), [typename](#) = [std::\\_\\_has\\_is\\_transparent\\_t](#)< [\\_Hash](#), [\\_Kt](#)>, [typename](#) = [std::\\_\\_has\\_is\\_transparent\\_t](#)< [\\_Pred](#), [\\_Kt](#)>>  
[std::pair](#)< [iterator](#), [iterator](#) > [equal\\_range](#) ([const](#) [\\_Kt](#) &\_\_k)
- [template](#)<[typename](#) [\\_Kt](#), [typename](#) = [std::\\_\\_has\\_is\\_transparent\\_t](#)< [\\_Hash](#), [\\_Kt](#)>, [typename](#) = [std::\\_\\_has\\_is\\_transparent\\_t](#)< [\\_Pred](#), [\\_Kt](#)>>  
[std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > [equal\\_range](#) ([const](#) [\\_Kt](#) &\_\_k) [const](#)

- `std::pair< iterator, iterator > equal_range` (const key\_type &\_\_key)
- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_key) const
- `_Base_iterator erase` (`_Base_const_iterator` \_\_it)
- `size_type erase` (const key\_type &\_\_key)
- `iterator erase` (`const_iterator` \_\_first, `const_iterator` \_\_last)
- `iterator erase` (`const_iterator` \_\_it)
- `iterator erase` (`iterator` \_\_it)
- `node_type extract` (const key\_type &\_\_key)
- `node_type extract` (`const_iterator` \_\_position)
- `template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>`  
`iterator find` (const \_Kt &\_\_k)
- `template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>`  
`const_iterator find` (const \_Kt &\_\_k) const
- `iterator find` (const key\_type &\_\_key)
- `const_iterator find` (const key\_type &\_\_key) const
- `template<typename _InputIterator>`  
`void insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `std::pair< iterator, bool > insert` (const value\_type &\_\_obj)
- `iterator insert` (`const_iterator` \_\_hint, const value\_type &\_\_obj)
- `iterator insert` (`const_iterator` \_\_hint, node\_type &&\_\_nh)
- `iterator insert` (`const_iterator` \_\_hint, value\_type &&\_\_obj)
- `insert_return_type insert` (node\_type &&\_\_nh)
- `void insert` (`std::initializer_list`< value\_type > \_\_l)
- `std::pair< iterator, bool > insert` (value\_type &&\_\_obj)
- `float max_load_factor` () const noexcept
- `void max_load_factor` (float \_\_f)
- `template<typename _H2, typename _P2>`  
`void merge` (`unordered_multiset`< \_Value, \_H2, \_P2, \_Alloc > &&\_\_source)
- `template<typename _H2, typename _P2>`  
`void merge` (`unordered_multiset`< \_Value, \_H2, \_P2, \_Alloc > &\_\_source)
- `template<typename _H2, typename _P2>`  
`void merge` (`unordered_set`< \_Value, \_H2, \_P2, \_Alloc > &&\_\_source)
- `template<typename _H2, typename _P2>`  
`void merge` (`unordered_set`< \_Value, \_H2, \_P2, \_Alloc > &\_\_source)
- `unordered_set & operator=` (const `unordered_set` &)=default
- `unordered_set & operator=` (`initializer_list`< value\_type > \_\_l)
- `unordered_set & operator=` (`unordered_set` &&)=default
- `void swap` (`unordered_set` &\_\_x) noexcept(noexcept(`declval`< `_Base` & >().swap(\_\_x)))

### Protected Member Functions

- constexpr void `_M_swap` (const `_Safe_container` &\_\_x) const noexcept

### Friends

- `template<typename _ItT, typename _SeqT, typename _CatT>`  
class `::__gnu_debug::Safe_iterator`
- `template<typename _ItT, typename _SeqT>`  
class `::__gnu_debug::Safe_local_iterator`

### 5.1003.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>,
typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >
```

Class std::unordered\_set with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

### 5.1004 std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_set>
```

#### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef \_Hashtable::pointer [pointer](#)
- typedef \_Hashtable::const\_pointer [const\\_pointer](#)
- typedef \_Hashtable::reference [reference](#)
- typedef \_Hashtable::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

#### Public Member Functions

- [unordered\\_set](#) ()=default
- template<typename \_InputIterator>  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator>  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator>  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator>  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_set](#) (const [allocator\\_type](#) &\_\_a)
- [unordered\\_set](#) (const [unordered\\_set](#) &)=default
- [unordered\\_set](#) (const [unordered\\_set](#) &\_\_uset, const [allocator\\_type](#) &\_\_a)
- [unordered\\_set](#) (initializer\_list< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- [unordered\\_set](#) (initializer\_list< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)

- **unordered\_set** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
  - **unordered\_set** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
  - **unordered\_set** (size\_type \_\_n, const allocator\_type &\_\_a)
  - **unordered\_set** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
  - **unordered\_set** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
  - **unordered\_set** (unordered\_set &&)=default
  - **unordered\_set** (unordered\_set &&\_\_uset, const allocator\_type &\_\_a) noexcept(noexcept(\_Hashtable(std::move(↵ \_\_uset.\_M\_h), \_\_a)))
  - size\_type **bucket** (const key\_type &\_\_key) const
  - size\_type **bucket\_count** () const noexcept
  - size\_type **bucket\_size** (size\_type \_\_n) const
  - const\_iterator **cbegin** () const noexcept
  - const\_iterator **cend** () const noexcept
  - void **clear** () noexcept
  - template<typename... \_Args>  
std::pair< iterator, bool > **emplace** (\_Args &&... \_\_args)
  - template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool **empty** () const noexcept
  - size\_type **erase** (const key\_type &\_\_x)
  - iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
  - allocator\_type **get\_allocator** () const noexcept
  - hasher **hash\_function** () const
  - template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void **insert** (initializer\_list< value\_type > \_\_l)
  - key\_equal key\_eq () const
  - float **load\_factor** () const noexcept
  - size\_type **max\_bucket\_count** () const noexcept
  - float **max\_load\_factor** () const noexcept
  - void **max\_load\_factor** (float \_\_z)
  - size\_type **max\_size** () const noexcept
  - unordered\_set & **operator=** (const unordered\_set &)=default
  - unordered\_set & **operator=** (initializer\_list< value\_type > \_\_l)
  - unordered\_set & **operator=** (unordered\_set &&)=default
  - void **rehash** (size\_type \_\_n)
  - void **reserve** (size\_type \_\_n)
  - size\_type **size** () const noexcept
  - void **swap** (unordered\_set &\_\_x) noexcept(noexcept(\_M\_h.swap(\_\_x.\_M\_h)))
- 
- iterator **begin** () noexcept
  - const\_iterator **begin** () const noexcept
- 
- iterator **end** () noexcept
  - const\_iterator **end** () const noexcept



- `std::pair< iterator, bool > insert (const value_type &__x)`
- `std::pair< iterator, bool > insert (value_type &&__x)`
  
- `iterator insert (const_iterator __hint, const value_type &__x)`
- `iterator insert (const_iterator __hint, value_type &&__x)`
  
- `iterator erase (const_iterator __position)`
- `iterator erase (iterator __position)`
  
- `iterator find (const key_type &__x)`
- `template<typename _Kt>`  
`auto find (const _Kt &__k) -> decltype(_M_h._M_find_tr(__k))`
- `const_iterator find (const key_type &__x) const`
- `template<typename _Kt>`  
`auto find (const _Kt &__k) const -> decltype(_M_h._M_find_tr(__k))`
  
- `size_type count (const key_type &__x) const`
- `template<typename _Kt>`  
`auto count (const _Kt &__k) const -> decltype(_M_h._M_count_tr(__k))`
  
- `bool contains (const key_type &__x) const`
- `template<typename _Kt>`  
`auto contains (const _Kt &__k) const -> decltype(_M_h._M_find_tr(__k), void(), true)`
  
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `template<typename _Kt>`  
`auto equal_range (const _Kt &__k) -> decltype(_M_h._M_equal_range_tr(__k))`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `template<typename _Kt>`  
`auto equal_range (const _Kt &__k) const -> decltype(_M_h._M_equal_range_tr(__k))`
  
- `local_iterator begin (size_type __n)`
- `const_local_iterator begin (size_type __n) const`
- `const_local_iterator cbegin (size_type __n) const`
  
- `local_iterator end (size_type __n)`
- `const_local_iterator end (size_type __n) const`
- `const_local_iterator cend (size_type __n) const`

## Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1>`  
`bool operator==(const unordered_set<_Value1, _Hash1, _Pred1, _Alloc1> &, const unordered_set<_Value1, _Hash1, _Pred1, _Alloc1> &)`

### 5.1004.1 Detailed Description

`template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc = allocator<_Value>>`  
**class** `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Since

C++11

#### Template Parameters

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>_Value</code> | Type of key objects.                                                              |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> . |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .                  |

Meets the requirements of a [container](#), and [unordered associative container](#)

Base is `_Hashtable`, dispatched at compile time via template alias `__uset_hashtable`.

### 5.1004.2 Member Typedef Documentation

#### `allocator_type`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::allocator_type
```

Public typedefs.

#### `const_iterator`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_iterator
```

Iterator-related typedefs.

#### `const_local_iterator`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_local_iterator
```

Iterator-related typedefs.

**const\_pointer**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::const_↵
pointer
```

Iterator-related typedefs.

**const\_reference**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::const_↵
reference
```

Iterator-related typedefs.

**difference\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::difference↵
_type
```

Iterator-related typedefs.

**hasher**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

**iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

**key\_equal**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

**key\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

**local\_iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
typedef _Hashtable::local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::local_iterator
```

Iterator-related typedefs.

### pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
typedef _Hashtable::pointer std::unordered_set<_Value, _Hash, _Pred, _Alloc>::pointer
```

Iterator-related typedefs.

### reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
typedef _Hashtable::reference std::unordered_set<_Value, _Hash, _Pred, _Alloc>::reference
```

Iterator-related typedefs.

### size\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
typedef _Hashtable::size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::size_type
```

Iterator-related typedefs.

### value\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
typedef _Hashtable::value_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::value_type
```

Public typedefs.

## 5.1004.3 Constructor & Destructor Documentation

### unordered\_set() [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::unordered_set<_Value, _Hash, _Pred, _Alloc>::unordered_set () [default]
```

Default constructor.

### unordered\_set() [2/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::unordered_set<_Value, _Hash, _Pred, _Alloc>::unordered_set (
    size_type __n,
    const hasher & __hf = hasher(),
    const key_equal & __eq = key_equal(),
    const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Default constructor creates no elements.

#### Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__n</code>  | Minimal initial number of buckets. |
| <code>__hf</code> | A hash functor.                    |

## Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__eq/</code> | A key equality functor. |
| <code>__a</code>   | An allocator object.    |

**unordered\_set()** [3/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    _InputIterator __first,
    _InputIterator __last,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq/ = key_equal(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_set from a range.

## Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eq/</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an unordered\_set consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is distance(`__first`,`__last`)).

**unordered\_set()** [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    const unordered_set< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

**unordered\_set()** [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    unordered_set< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

**unordered\_set()** [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    const allocator_type & __a) [inline], [explicit]
```

Creates an unordered\_set with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

#### unordered\_set() [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    initializer_list< value_type > __l,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_set from an initializer\_list.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_set consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

### 5.1004.4 Member Function Documentation

#### begin() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

#### begin() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

#### begin() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
    size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| $\leftarrow$     | The bucket index. |
| <code>__n</code> |                   |

**Returns**

A read-only local iterator.

**begin() [4/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| $\leftarrow$     | The bucket index. |
| <code>__n</code> |                   |

**Returns**

A read-only local iterator.

**bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_set.

**cbegin() [1/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

**cbegin() [2/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| $\leftarrow$     | The bucket index. |
| <code>__n</code> |                   |



**Returns**

A read-only local iterator.

**cend()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

**cend()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

**clear()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an `unordered_set`. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::contains (
    const _Kt & __k) const -> decltype(_M_h._M_find_tr(__k), void(), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

**Returns**



True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set<_Value, _Hash, _Pred, _Alloc >::contains (
    const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                                                                                   |                                |
|-----------------------------------------------------------------------------------|--------------------------------|
|  | Key of elements to be located. |
|  |                                |

**Returns**



True if there is any element with the specified key.

**count()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set<_Value, _Hash, _Pred, _Alloc >::count (
    const _Kt & __k) const -> decltype(_M_h._M_count_tr(__k)) [inline]
```

Finds the number of elements.

**Parameters**

|                                                                                     |                     |
|-------------------------------------------------------------------------------------|---------------------|
|  | Element to located. |
|  |                     |

**Returns**

Number of elements with specified key.



This function only makes sense for unordered\_multisets; for unordered\_set the result will either be 0 (not present) or 1 (present).

**count()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc >::count (
    const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                                                                                     |                     |
|-------------------------------------------------------------------------------------|---------------------|
|  | Element to located. |
|  |                     |

**Returns**

Number of elements with specified key.

This function only makes sense for unordered\_multisets; for unordered\_set the result will either be 0 (not present) or 1 (present).

**emplace()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace (
    _Args &&... __args) [inline]
```

Attempts to build and insert an element into the `unordered_set`.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a `bool` that is true if the element was actually inserted.

This function attempts to build and insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`. Insertion requires amortized constant time.

**emplace\_hint()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

**Returns**

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant time.

**empty()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set< _Value, _Hash, _Pred, _Alloc >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the `unordered_set` is empty.

**end()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

**end()** [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

**end()** [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::end (
    size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|              |                   |
|--------------|-------------------|
| $\leftarrow$ | The bucket index. |
| $n$          |                   |

**Returns**

A read-only local iterator.

**end()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::end (
    size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|              |                   |
|--------------|-------------------|
| $\leftarrow$ | The bucket index. |
| $n$          |                   |

**Returns**

A read-only local iterator.

**equal\_range()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
```

```
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range (
    const _Kt & __k) -> decltype(_M_h._M_equal_range_tr(__k))    [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↔</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

**equal\_range() [2/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range (
    const _Kt & __k) const -> decltype(_M_h._M_equal_range_tr(__k))    [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↔</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

**equal\_range() [3/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, iterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range
(
    const key_type & __x)    [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↔</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

**equal\_range()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< const_iterator, const_iterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc >↵
::equal_range (
    const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

## Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__key</code> | Key to be located. |
| <code>__x</code>   |                    |

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

**erase()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
    const key_type & __x) [inline]
```

Erases elements according to the provided key.

## Parameters

|                    |                              |
|--------------------|------------------------------|
| <code>__key</code> | Key of element to be erased. |
| <code>__x</code>   |                              |

## Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered\_set. For an unordered\_set the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an unordered\_set.

## Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

## Returns

The iterator `__last`.

This function erases a sequence of elements from an unordered\_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.



**erase()** [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
    const_iterator __position) [inline]
```

Erases an element from an unordered\_set.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
    iterator __position) [inline]
```

Erases an element from an unordered\_set.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
    const _Kt & __k) -> decltype(_M_h._M_find_tr(__k)) [inline]
```

Tries to locate an element in an unordered\_set.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find() [2/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
    const _Kt & __k) const -> decltype(_M_h._M_find_tr(__k))    [inline]
```

Tries to locate an element in an unordered\_set.

**Parameters**

|                                                                                   |                        |
|-----------------------------------------------------------------------------------|------------------------|
|  | Element to be located. |
|  |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find() [3/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
    const key_type & __x) [inline]
```

Tries to locate an element in an unordered\_set.

**Parameters**

|                                                                                     |                        |
|-------------------------------------------------------------------------------------|------------------------|
|  | Element to be located. |
|  |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find() [4/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
    const key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_set.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**get\_allocator()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Returns the allocator object used by the `unordered_set`.

**hash\_function()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the `unordered_set` was constructed.

**insert()** [1/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert()** [2/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    const value_type & __x) [inline]
```

Attempts to insert an element into the `unordered_set`.

## Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

**insert()** [3/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    const value_type & __x) [inline]
```

Attempts to insert an element into the unordered\_set.

## Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

## Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

**insert()** [4/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    value_type && __x) [inline]
```

Attempts to insert an element into the unordered\_set.

## Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

**insert()** [5/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the `unordered_set`.

**Parameters**

|   |                                                                                    |
|---|------------------------------------------------------------------------------------|
| ↔ | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
| ↔ |                                                                                    |
| ↔ |                                                                                    |
| ↔ |                                                                                    |
| / |                                                                                    |

Complexity similar to that of the range constructor.

**insert()** [6/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    value_type && __x) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

|   |                         |
|---|-------------------------|
| ↔ | Element to be inserted. |
| ↔ |                         |
| ↔ |                         |
| ↔ |                         |
| / |                         |

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a `bool` that is true if the element was actually inserted.

This function attempts to insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`.

Insertion requires amortized constant time.

**key\_eq()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the `unordered_set` was constructed.

**load\_factor()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::load_factor () const [inline], [noexcept]
```

Returns the average number of elements per bucket.

**max\_bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const [inline],
[noexcept]
```

Returns the maximum number of buckets of the unordered\_set.

**max\_load\_factor()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_set tries to keep the load factor less than or equal to.

**max\_load\_factor()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
    float __z) [inline]
```

Change the unordered\_set maximum load factor.

**Parameters**

|                                                                                     |                              |
|-------------------------------------------------------------------------------------|------------------------------|
|  | The new maximum load factor. |
| <b>_z</b>                                                                           |                              |

**max\_size()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the unordered\_set.

**operator=()** [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
    const unordered_set< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy assignment operator.

**operator=()** [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
    initializer_list< value_type > __l) [inline]
```

Unordered\_set list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills an unordered\_set with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the unordered\_set and that the resulting unordered\_set's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
    unordered_set< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move assignment operator.

**rehash()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::rehash (
    size_type __n) [inline]
```

May rehash the unordered\_set.

**Parameters**

|   |                            |
|---|----------------------------|
| ↩ | The new number of buckets. |
| n |                            |

Rehash will occur only if the new number of buckets respect the unordered\_set maximum load factor.

**reserve()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reserve (
    size_type __n) [inline]
```

Prepare the unordered\_set for a specified number of elements.

**Parameters**

|   |                              |
|---|------------------------------|
| ↩ | Number of elements required. |
| n |                              |

Same as `rehash(ceil(n / max_load_factor()))`.

### size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the `unordered_set`.

### swap()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap (
    unordered_set< _Value, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another `unordered_set`.

#### Parameters

|                  |                                                                        |
|------------------|------------------------------------------------------------------------|
| <code>__x</code> | An <code>unordered_set</code> of the same element and allocator types. |
|------------------|------------------------------------------------------------------------|

This exchanges the elements between two sets in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 5.1005 std::pmr::unsynchronized\_pool\_resource Class Reference

```
#include <memory_resource>
```

Inheritance diagram for `std::pmr::unsynchronized_pool_resource`:





### Public Member Functions

- **unsynchronized\_pool\_resource** (const [pool\\_options](#) &\_\_opts)
- **unsynchronized\_pool\_resource** (const [pool\\_options](#) &\_\_opts, [memory\\_resource](#) \* \_\_upstream)
- **unsynchronized\_pool\_resource** (const [unsynchronized\\_pool\\_resource](#) &)=delete
- **unsynchronized\_pool\_resource** ([memory\\_resource](#) \* \_\_upstream)
- void \* **allocate** (size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- void **deallocate** (void \* \_\_p, size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- bool **is\_equal** (const [memory\\_resource](#) & \_\_other) const noexcept
- [unsynchronized\\_pool\\_resource](#) & **operator=** (const [unsynchronized\\_pool\\_resource](#) &)=delete
- [pool\\_options](#) **options** () const noexcept
- void **release** ()
- [memory\\_resource](#) \* **upstream\_resource** () const noexcept

### Protected Member Functions

- void \* **do\_allocate** (size\_t \_\_bytes, size\_t \_\_alignment) override
- void **do\_deallocate** (void \* \_\_p, size\_t \_\_bytes, size\_t \_\_alignment) override
- bool **do\_is\_equal** (const [memory\\_resource](#) & \_\_other) const noexcept override

#### 5.1005.1 Detailed Description

A non-thread-safe memory resource that manages pools of fixed-size blocks.

Since

C++17

#### 5.1005.2 Member Function Documentation

##### **do\_allocate()**

```
void * std::pmr::unsynchronized_pool_resource::do_allocate (
    size_t __bytes,
    size_t __alignment) [override], [protected], [virtual]
```

Implements [std::pmr::memory\\_resource](#).

##### **do\_deallocate()**

```
void std::pmr::unsynchronized_pool_resource::do_deallocate (
    void * __p,
    size_t __bytes,
    size_t __alignment) [override], [protected], [virtual]
```

Implements [std::pmr::memory\\_resource](#).

##### **do\_is\_equal()**

```
bool std::pmr::unsynchronized_pool_resource::do_is_equal (
    const memory\_resource & __other) const [inline], [override], [protected], [virtual],
[noexcept]
```

Implements [std::pmr::memory\\_resource](#).

The documentation for this class was generated from the following file:

- [memory\\_resource](#)

## 5.1006 `std::uses_allocator< typename, typename >` Struct Template Reference

### 5.1006.1 Detailed Description

```
template<typename, typename>
struct std::uses_allocator< typename, typename >
```

Declare `uses_allocator` so it can be specialized in `<queue>` etc.  
 The documentation for this struct was generated from the following file:

- [memoryfwd.h](#)

## 5.1007 `std::uses_allocator< tuple< _Types... >, _Alloc >` Struct Template Reference

```
#include <tuple>
```

### 5.1007.1 Detailed Description

```
template<typename... _Types, typename _Alloc>
struct std::uses_allocator< tuple< _Types... >, _Alloc >
```

Partial specialization for tuples.  
 The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.1008 `std::chrono::utc_clock` Class Reference

```
#include <chrono>
```

### Public Types

- using `duration`
- using `period`
- using `rep`
- using `time_point`

### Static Public Member Functions

- `template<typename _Duration>`  
`static utc\_time< common\_type\_t< _Duration, seconds > > from\_sys (const sys\_time< _Duration > &__t)`
- `static time\_point now ()`
- `template<typename _Duration>`  
`static sys\_time< common\_type\_t< _Duration, seconds > > to\_sys (const utc\_time< _Duration > &__t)`

### Static Public Attributes

- static constexpr bool `is_steady`

### 5.1008.1 Detailed Description

A clock that measures Universal Coordinated Time (UTC).  
 The epoch is 1970-01-01 00:00:00.

Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

## 5.1009 std::valarray<\_Tp> Class Template Reference

```
#include <valarray>
```

### Public Types

- typedef `_Tp` **value\_type**

### Public Member Functions

- `valarray` () noexcept
- template<class `_Dom`>  
`valarray` (const `_Expr`< `_Dom`, `_Tp` > &`__e`)
- `valarray` (const `_Tp` &, `size_t`)
- template<typename `_Tp`>  
`valarray` (const `_Tp` \*`__restrict` `__p`, `size_t` `__n`)
- `valarray` (const `_Tp` \*`__restrict` `__`, `size_t`)
- `valarray` (const `gslice_array`< `_Tp` > &)
- `valarray` (const `indirect_array`< `_Tp` > &)
- `valarray` (const `mask_array`< `_Tp` > &)
- `valarray` (const `slice_array`< `_Tp` > &)
- `valarray` (const `valarray` &)
- `valarray` (`initializer_list`< `_Tp` >)
- `valarray` (`size_t`)
- `valarray` (`valarray` &&) noexcept
- `_Expr`< `_ValFunClos`< `_ValArray`, `_Tp` >, `_Tp` > `apply` (`_Tp` `__func`(`_Tp`)) const
- `_Expr`< `_RefFunClos`< `_ValArray`, `_Tp` >, `_Tp` > `apply` (`_Tp` `__func`(const `_Tp` &)) const
- `valarray`< `_Tp` > `cshift` (int `__n`) const
- `_Tp` `max` () const
- `_Tp` `min` () const
- `_UnaryOp`< `__logical_not` >::`Rt operator!` () const
- template<class `_Dom`>  
`valarray`< `_Tp` > & `operator%=` (const `_Expr`< `_Dom`, `_Tp` > &)
- `valarray`< `_Tp` > & `operator%=` (const `_Tp` &)
- `valarray`< `_Tp` > & `operator%=` (const `valarray`< `_Tp` > &)
- template<class `_Dom`>  
`valarray`< `_Tp` > & `operator&=` (const `_Expr`< `_Dom`, `_Tp` > &)
- `valarray`< `_Tp` > & `operator&=` (const `_Tp` &)
- `valarray`< `_Tp` > & `operator&=` (const `valarray`< `_Tp` > &)
- template<class `_Dom`>  
`valarray`< `_Tp` > & `operator*=` (const `_Expr`< `_Dom`, `_Tp` > &)
- `valarray`< `_Tp` > & `operator*=` (const `_Tp` &)
- `valarray`< `_Tp` > & `operator*=` (const `valarray`< `_Tp` > &)
- `_UnaryOp`< `__unary_plus` >::`Rt operator+` () const
- template<class `_Dom`>  
`valarray`< `_Tp` > & `operator+=` (const `_Expr`< `_Dom`, `_Tp` > &)
- `valarray`< `_Tp` > & `operator+=` (const `_Tp` &)
- `valarray`< `_Tp` > & `operator+=` (const `valarray`< `_Tp` > &)
- `_UnaryOp`< `__negate` >::`Rt operator-` () const
- template<class `_Dom`>  
`valarray`< `_Tp` > & `operator-=` (const `_Expr`< `_Dom`, `_Tp` > &)
- `valarray`< `_Tp` > & `operator-=` (const `_Tp` &)

- `valarray<_Tp> & operator=` (const `valarray<_Tp>` &)
- `template<class _Dom>`  
`valarray<_Tp> & operator/=` (const `_Expr<_Dom, _Tp>` &)
- `valarray<_Tp> & operator/=` (const `_Tp` &)
- `valarray<_Tp> & operator/=` (const `valarray<_Tp>` &)
- `template<class _Dom>`  
`valarray<_Tp> & operator<<=` (const `_Expr<_Dom, _Tp>` &)
- `valarray<_Tp> & operator<<=` (const `_Tp` &)
- `valarray<_Tp> & operator<<=` (const `valarray<_Tp>` &)
- `template<class _Dom>`  
`valarray<_Tp> & operator=` (const `_Expr<_Dom, _Tp>` &)
- `valarray<_Tp> & operator=` (const `_Tp` &\_\_t)
- `valarray<_Tp> & operator=` (const `gslice_array<_Tp>` &\_\_ga)
- `valarray<_Tp> & operator=` (const `indirect_array<_Tp>` &\_\_ia)
- `valarray<_Tp> & operator=` (const `mask_array<_Tp>` &\_\_ma)
- `valarray<_Tp> & operator=` (const `slice_array<_Tp>` &\_\_sa)
- `valarray<_Tp> & operator=` (const `valarray<_Tp>` &\_\_v)
- `valarray & operator=` (`initializer_list<_Tp>` \_\_l)
- `valarray<_Tp> & operator=` (`valarray<_Tp>` &&\_\_v) noexcept
- `template<class _Dom>`  
`valarray<_Tp> & operator>>=` (const `_Expr<_Dom, _Tp>` &)
- `valarray<_Tp> & operator>>=` (const `_Tp` &)
- `valarray<_Tp> & operator>>=` (const `valarray<_Tp>` &)
- `gslice_array<_Tp> operator[]` (const `gslice` &\_\_s)
- `_Expr<_GClos<_ValArray, _Tp>, _Tp> operator[]` (const `gslice` &\_\_s) const
- `mask_array<_Tp> operator[]` (const `valarray<bool>` &\_\_m)
- `valarray<_Tp> operator[]` (const `valarray<bool>` &\_\_m) const
- `indirect_array<_Tp> operator[]` (const `valarray<size_t>` &\_\_i)
- `_Expr<_IClos<_ValArray, _Tp>, _Tp> operator[]` (const `valarray<size_t>` &\_\_i) const
- `_Tp & operator[]` (`size_t` \_\_i) noexcept
- `const _Tp & operator[]` (`size_t`) const noexcept
- `slice_array<_Tp> operator[]` (`slice` \_\_s)
- `_Expr<_SClos<_ValArray, _Tp>, _Tp> operator[]` (`slice` \_\_s) const
- `template<class _Dom>`  
`valarray<_Tp> & operator^=` (const `_Expr<_Dom, _Tp>` &)
- `valarray<_Tp> & operator^=` (const `_Tp` &)
- `valarray<_Tp> & operator^=` (const `valarray<_Tp>` &)
- `template<class _Dom>`  
`valarray<_Tp> & operator|=` (const `_Expr<_Dom, _Tp>` &)
- `valarray<_Tp> & operator|=` (const `_Tp` &)
- `valarray<_Tp> & operator|=` (const `valarray<_Tp>` &)
- `_UnaryOp<__bitwise_not>::Rt operator~` () const
- `void resize` (`size_t` \_\_size, `_Tp` \_\_c=\_Tp())
- `valarray<_Tp> shift` (`int` \_\_n) const
- `size_t size` () const
- `_Tp sum` () const
- `void swap` (`valarray<_Tp>` &\_\_v) noexcept

## Friends

- `struct _Array<_Tp>`

### 5.1009.1 Detailed Description

```
template<class _Tp>
class std::valarray<_Tp>
```

Smart array designed to support numeric processing.

A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

#### Template Parameters

|                  |                              |
|------------------|------------------------------|
| <code>_Tp</code> | Type of object in the array. |
|------------------|------------------------------|

### 5.1009.2 Constructor & Destructor Documentation

#### `valarray()`

```
template<class _Tp>
std::valarray<_Tp>::valarray (
    const _Tp * __restrict__,
    size_t )
```

Construct an array initialized to the first  $n$  elements of  $t$ .

### 5.1009.3 Member Function Documentation

#### `operator"!")()`

```
template<class _Tp>
_UnaryOp< __logical_not >::_Rt std::valarray<_Tp>::operator!  () const
```

Return a new valarray by applying unary ! to each element.

#### `operator%=( ) [1/2]`

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator%= (
    const _Tp & )
```

Set each element  $e$  of array to  $e \% t$ .

#### `operator%=( ) [2/2]`

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator%= (
    const valarray<_Tp> & )
```

Modulo elements of array by corresponding elements of  $v$ .

#### `operator&=( ) [1/2]`

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator&= (
    const _Tp & )
```

Set each element  $e$  of array to  $e \& t$ .

**operator&=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
    const valarray< _Tp > & )
```

Logical and corresponding elements of *v* with elements of array.

**operator\*=()** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
    const _Tp & )
```

Multiply each element of array by *t*.

**operator\*=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
    const valarray< _Tp > & )
```

Multiply elements of array by corresponding elements of *v*.

**operator+()**

```
template<class _Tp>
_UnaryOp< __unary_plus >::_Rt std::valarray< _Tp >::operator+ () const
```

Return a new valarray by applying unary + to each element.

**operator+=()** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
    const _Tp & )
```

Add *t* to each element of array.

**operator+=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
    const valarray< _Tp > & )
```

Add corresponding elements of *v* to elements of array.

**operator-()**

```
template<class _Tp>
_UnaryOp< __negate >::_Rt std::valarray< _Tp >::operator- () const
```

Return a new valarray by applying unary - to each element.

**operator-=()** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
    const _Tp & )
```

Subtract *t* to each element of array.

**operator-=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
    const valarray< _Tp > & )
```

Subtract corresponding elements of *v* from elements of array.

**operator/=(** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
    const _Tp & )
```

Divide each element of array by *t*.

**operator/=(** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
    const valarray< _Tp > & )
```

Divide elements of array by corresponding elements of *v*.

**operator<<=(** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
    const _Tp & )
```

Left shift each element *e* of array by *t* bits.

**operator<<=(** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
    const valarray< _Tp > & )
```

Left shift elements of array by corresponding elements of *v*.

**operator>>=(** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
    const _Tp & )
```

Right shift each element *e* of array by *t* bits.

**operator>>=(** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
    const valarray< _Tp > & )
```

Right shift elements of array by corresponding elements of *v*.

**operator^=(** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
    const _Tp & )
```

Set each element *e* of array to  $e \wedge t$ .

**operator^=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
    const valarray< _Tp > & )
```

Logical xor corresponding elements of *v* with elements of array.

**operator" |=()** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
    const _Tp & )
```

Set each element *e* of array to *e* | *t*.

**operator" |=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
    const valarray< _Tp > & )
```

Logical or corresponding elements of *v* with elements of array.

**operator~()**

```
template<class _Tp>
_UnaryOp< __bitwise_not >::Rt std::valarray< _Tp >::operator~ () const
```

Return a new valarray by applying unary ~ to each element.

The documentation for this class was generated from the following file:

- [valarray](#)

**5.1010 `std::__debug::vector<_Tp, _Allocator>` Class Template Reference**

```
#include <vector>
```

Inheritance diagram for `std::__debug::vector<_Tp, _Allocator>`:

**Public Types**

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator<_Base_const_iterator, vector>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**



- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, vector>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **vector** (`_Base_ref __x`)
- template<class `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
constexpr **vector** (`_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator()`)
- constexpr **vector** (`const _Allocator &__a`) noexcept
- **vector** (`const vector &`)=default
- constexpr **vector** (`const vector &__x, const __type_identity_t< allocator_type > &__a`)
- constexpr **vector** (`initializer_list< value_type > __l, const allocator_type &__a=allocator_type()`)
- constexpr **vector** (`size_type __n, const __type_identity_t<_Tp> &__value, const _Allocator &__a=_Allocator()`)
- constexpr **vector** (`size_type __n, const _Allocator &__a=_Allocator()`)
- **vector** (`vector &&`)=default
- constexpr **vector** (`vector &&__x, const __type_identity_t< allocator_type > &__a`) noexcept(`std::is_nothrow_constructible<_Base, _Base, const allocator_type &>::value`)
- constexpr const `_Base & _M_base` () const noexcept
- constexpr `_Base & _M_base` () noexcept
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
constexpr void **assign** (`_InputIterator __first, _InputIterator __last`)
- constexpr void **assign** (`initializer_list< value_type > __l`)
- constexpr void **assign** (`size_type __n, const _Tp &__u`)
- constexpr const\_reference **back** () const noexcept
- constexpr reference **back** () noexcept
- constexpr const\_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr size\_type **capacity** () const noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr void **clear** () noexcept
- constexpr const\_reverse\_iterator **crbegin** () const noexcept
- constexpr const\_reverse\_iterator **crend** () const noexcept
- template<typename... `_Args`>  
constexpr iterator **emplace** (`const_iterator __position, _Args &&... __args`)
- template<typename... `_Args`>  
constexpr reference **emplace\_back** (`_Args &&... __args`)
- constexpr const\_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator **erase** (`const_iterator __first, const_iterator __last`)
- constexpr iterator **erase** (`const_iterator __position`)
- constexpr const\_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- template<class `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
constexpr iterator **insert** (`const_iterator __position, _InputIterator __first, _InputIterator __last`)

- template<typename \_Up = \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<!std::\_\_are\_same< \_Up, bool >::\_\_value, iterator >::\_\_type insert  
(const\_iterator \_\_position, \_Tp && \_\_x)
- constexpr iterator insert (const\_iterator \_\_position, const \_Tp & \_\_x)
- constexpr iterator insert (const\_iterator \_\_position, initializer\_list< value\_type > \_\_l)
- constexpr iterator insert (const\_iterator \_\_position, size\_type \_\_n, const \_Tp & \_\_x)
- vector & operator= (const vector &)=default
- constexpr vector & operator= (initializer\_list< value\_type > \_\_l)
- vector & operator= (vector &&)=default
- constexpr const\_reference operator[] (size\_type \_\_n) const noexcept
- constexpr reference operator[] (size\_type \_\_n) noexcept
- constexpr void pop\_back () noexcept
- template<typename \_Up = \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<!std::\_\_are\_same< \_Up, bool >::\_\_value, void >::\_\_type push\_back (\_Tp  
&& \_\_x)
- constexpr void push\_back (const \_Tp & \_\_x)
- constexpr const\_reverse\_iterator rbegin () const noexcept
- constexpr reverse\_iterator rbegin () noexcept
- constexpr const\_reverse\_iterator rend () const noexcept
- constexpr reverse\_iterator rend () noexcept
- constexpr void reserve (size\_type \_\_n)
- constexpr void resize (size\_type \_\_sz)
- constexpr void resize (size\_type \_\_sz, const \_Tp & \_\_c)
- constexpr void shrink\_to\_fit ()
- constexpr void swap (vector & \_\_x) noexcept(/\*conditional \*/)

### Protected Member Functions

- bool \_M\_requires\_reallocation (size\_type \_\_elements) const noexcept
- constexpr void \_M\_swap (const \_Safe\_container & \_\_x) const noexcept
- constexpr void \_M\_update\_guaranteed\_capacity () noexcept

### Protected Attributes

- size\_type \_M\_guaranteed\_capacity

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug::\_Safe\_iterator

#### 5.1010.1 Detailed Description

template<typename \_Tp, typename \_Allocator = std::allocator<\_Tp>>  
class std::\_\_debug::vector< \_Tp, \_Allocator >

Class std::vector with safety/checking/debug instrumentation.

### 5.1010.2 Constructor & Destructor Documentation

#### vector()

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
std::__debug::vector< _Tp, _Allocator >::vector (
    _Base_ref __x) [inline], [constexpr]
```

Construction from a normal-mode vector.

The documentation for this class was generated from the following file:

- [debug/vector](#)

### 5.1011 std::vector< \_Tp, \_Alloc > Class Template Reference

```
#include <vector>
```

Inheritance diagram for std::vector< \_Tp, \_Alloc >:



#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, vector >` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- [vector](#) ()=default
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr [vector](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- constexpr [vector](#) (const allocator\_type &\_\_a) noexcept
- constexpr [vector](#) (const [vector](#) &\_\_x)
- constexpr [vector](#) (const [vector](#) &\_\_x, const \_\_type\_identity\_t< allocator\_type > &\_\_a)
- constexpr [vector](#) (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- constexpr [vector](#) (size\_type \_\_n, const allocator\_type &\_\_a=allocator\_type())
- constexpr [vector](#) (size\_type \_\_n, const value\_type &\_\_value, const allocator\_type &\_\_a=allocator\_type())
- [vector](#) ([vector](#) &&) noexcept=default
- constexpr [vector](#) ([vector](#) &&\_\_rv, const \_\_type\_identity\_t< allocator\_type > &\_\_m) noexcept(noexcept([vector](#)(std::declval<  
[vector](#) && >(), std::declval< const allocator\_type & >(), std::declval< typename \_Alloc\_traits::is\_always\_equal  
>())))
- constexpr ~[vector](#) () noexcept
- template<typename... \_Args>  
constexpr auto [M\\_emplace\\_aux](#) (const\_iterator \_\_position, \_Args &&... \_\_args) -> iterator
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- constexpr void [assign](#) (initializer\_list< value\_type > \_\_l)
- constexpr void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- constexpr reference [at](#) (size\_type \_\_n)
- constexpr const\_reference [at](#) (size\_type \_\_n) const
- constexpr const\_reference [back](#) () const noexcept
- constexpr reference [back](#) () noexcept
- constexpr const\_iterator [begin](#) () const noexcept
- constexpr iterator [begin](#) () noexcept
- constexpr size\_type [capacity](#) () const noexcept
- constexpr const\_iterator [cbegin](#) () const noexcept
- constexpr const\_iterator [cend](#) () const noexcept
- constexpr void [clear](#) () noexcept
- constexpr const\_reverse\_iterator [crbegin](#) () const noexcept
- constexpr const\_reverse\_iterator [crend](#) () const noexcept
- constexpr const \_Tp \* [data](#) () const noexcept
- constexpr \_Tp \* [data](#) () noexcept
- template<typename... \_Args>  
constexpr iterator [emplace](#) (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
constexpr reference [emplace\\_back](#) (\_Args &&... \_\_args)
- constexpr bool [empty](#) () const noexcept
- constexpr const\_iterator [end](#) () const noexcept
- constexpr iterator [end](#) () noexcept
- constexpr iterator [erase](#) (const\_iterator \_\_first, const\_iterator \_\_last)
- constexpr iterator [erase](#) (const\_iterator \_\_position)
- constexpr const\_reference [front](#) () const noexcept
- constexpr reference [front](#) () noexcept
- constexpr allocator\_type [get\\_allocator](#) () const noexcept
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr iterator [insert](#) (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- constexpr iterator [insert](#) (const\_iterator \_\_position, const value\_type &\_\_x)
- constexpr iterator [insert](#) (const\_iterator \_\_position, initializer\_list< value\_type > \_\_l)

- constexpr iterator `insert` (const\_iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- constexpr iterator `insert` (const\_iterator \_\_position, value\_type &&\_\_x)
- constexpr size\_type `max_size` () const noexcept
- constexpr `vector` & `operator=` (const `vector` &\_\_x)
- constexpr `vector` & `operator=` (initializer\_list< value\_type > \_\_l)
- constexpr `vector` & `operator=` (`vector` &&\_\_x) noexcept(\_Alloc\_traits::\_S\_nothrow\_move())
- constexpr const\_reference `operator[]` (size\_type \_\_n) const noexcept
- constexpr reference `operator[]` (size\_type \_\_n) noexcept
- constexpr void `pop_back` () noexcept
- constexpr void `push_back` (const value\_type &\_\_x)
- constexpr void `push_back` (value\_type &&\_\_x)
- constexpr `const_reverse_iterator` `rbegin` () const noexcept
- constexpr `reverse_iterator` `rbegin` () noexcept
- constexpr `const_reverse_iterator` `rend` () const noexcept
- constexpr `reverse_iterator` `rend` () noexcept
- constexpr void `reserve` (size\_type \_\_n)
- constexpr void `resize` (size\_type \_\_new\_size)
- constexpr void `resize` (size\_type \_\_new\_size, const value\_type &\_\_x)
- constexpr void `shrink_to_fit` ()
- constexpr size\_type `size` () const noexcept
- constexpr void `swap` (`vector` &\_\_x) noexcept

### Protected Member Functions

- constexpr pointer `_M_allocate` (size\_t \_\_n)
- template<typename \_ForwardIterator>  
constexpr pointer `_M_allocate_and_copy` (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_ForwardIterator>  
constexpr void `_M_assign_aux` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, std::forward\_iterator\_tag)
- template<typename \_InputIterator>  
constexpr void `_M_assign_aux` (\_InputIterator \_\_first, \_InputIterator \_\_last, std::input\_iterator\_tag)
- template<typename \_InputIterator>  
constexpr void `_M_assign_dispatch` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_Integer>  
constexpr void `_M_assign_dispatch` (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- constexpr size\_type `_M_check_len` (size\_type \_\_n, const char \*\_\_s) const
- constexpr void `_M_create_storage` (size\_t \_\_n)
- constexpr void `_M_deallocate` (pointer \_\_p, size\_t \_\_n)
- constexpr void `_M_default_append` (size\_type \_\_n)
- constexpr void `_M_default_initialize` (size\_type \_\_n)
- template<typename... \_Args>  
constexpr iterator `_M_emplace_aux` (const\_iterator \_\_position, \_Args &&... \_\_args)
- constexpr iterator `_M_emplace_aux` (const\_iterator \_\_position, value\_type &&\_\_v)
- constexpr iterator `_M_erase` (iterator \_\_first, iterator \_\_last)
- constexpr iterator `_M_erase` (iterator \_\_position)
- constexpr void `_M_erase_at_end` (pointer \_\_pos) noexcept
- constexpr void `_M_fill_append` (size\_type \_\_n, const value\_type &\_\_x)
- constexpr void `_M_fill_assign` (size\_type \_\_n, const value\_type &\_\_val)
- constexpr void `_M_fill_initialize` (size\_type \_\_n, const value\_type &\_\_value)
- constexpr void `_M_fill_insert` (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- constexpr const \_Tp\_alloc\_type & `_M_get_Tp_allocator` () const noexcept

- `constexpr _Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `template<typename _Arg>`  
`constexpr void _M_insert_aux (iterator __position, _Arg &&__arg)`
- `template<typename _InputIterator>`  
`constexpr void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _Integer>`  
`constexpr void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __val, __true_type)`
- `constexpr iterator _M_insert_rval (const_iterator __position, value_type &&__v)`
- `constexpr void _M_range_check (size_type __n) const`
- `template<typename _ForwardIterator>`  
`constexpr void _M_range_initialize ( _ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator>`  
`constexpr void _M_range_initialize ( _InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _Iterator, typename _Sentinel>`  
`constexpr void _M_range_initialize_n ( _Iterator __first, _Sentinel __last, size_type __n)`
- `template<typename _ForwardIterator>`  
`constexpr void _M_range_insert (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator>`  
`constexpr void _M_range_insert (iterator __pos, _InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename... _Args>`  
`constexpr void _M_realloc_append ( _Args &&... __args)`
- `template<typename... _Args>`  
`constexpr void _M_realloc_insert (iterator __position, _Args &&... __args)`
- `constexpr bool _M_shrink_to_fit ()`

### Static Protected Member Functions

- `static constexpr size_type _S_check_init_len (size_type __n, const allocator_type &__a)`
- `static constexpr size_type _S_max_size (const _Tp_alloc_type &__a) noexcept`

### Protected Attributes

- `_Vector_impl _M_impl`

#### 5.1011.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
**class** `std::vector<_Tp, _Alloc>`

A standard container which offers fixed time access to individual elements in any order.

Since

C++98

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

### 5.1011.2 Constructor & Destructor Documentation

#### vector() [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector () [default]
```

Creates a vector with no elements.

Referenced by [insert\(\)](#), and [operator=\(\)](#).

#### vector() [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    const allocator_type & __a) [inline], [explicit], [constexpr], [noexcept]
```

Creates a vector with no elements.

##### Parameters

|                                                                                                                       |                      |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__a</code> | An allocator object. |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|

#### vector() [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    size_type __n,
    const allocator_type & __a = allocator_type()) [inline], [explicit], [constexpr]
```

Creates a vector with default constructed elements.

##### Parameters

|                                                                                                                         |                                             |
|-------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| <a href="#"></a><br><code>__n</code> | The number of elements to initially create. |
| <a href="#"></a><br><code>__a</code> | An allocator.                               |

This constructor fills the vector with `__n` default constructed elements.

#### vector() [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    size_type __n,
    const value_type & __value,
    const allocator_type & __a = allocator_type()) [inline], [constexpr]
```

Creates a vector with copies of an exemplar element.

##### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator.                               |

This constructor fills the vector with `__n` copies of `__value`.

**vector()** [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
    const vector<_Tp, _Alloc> & __x) [inline], [constexpr]
```

Vector copy constructor.

**Parameters**

|       |                                                    |
|-------|----------------------------------------------------|
| $\_x$ | A vector of identical element and allocator types. |
|-------|----------------------------------------------------|

All the elements of  $\_x$  are copied, but any unused capacity in  $\_x$  will not be copied (i.e. `capacity() == size()` in the new vector).

The newly-created vector uses a copy of the allocator object used by  $\_x$  (unless the allocator traits dictate a different object).

**vector()** [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
    vector<_Tp, _Alloc> && ) [default], [noexcept]
```

Vector move constructor.

The newly-created vector contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified vector.

**vector()** [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
    const vector<_Tp, _Alloc> & __x,
    const __type_identity_t< allocator_type > & __a) [inline], [constexpr]
```

Copy constructor with alternative allocator.

**vector()** [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
    vector<_Tp, _Alloc> && __rv,
    const __type_identity_t< allocator_type > & __m) [inline], [constexpr], [noexcept]
```

Move constructor with alternative allocator.

**vector()** [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
    initializer_list< value_type > __l,
    const allocator_type & __a = allocator_type()) [inline], [constexpr]
```

Builds a vector from an initializer list.

**Parameters**

|       |                      |
|-------|----------------------|
| $\_l$ | An initializer_list. |
| $\_a$ | An allocator.        |



Create a vector consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor  $N$  times (where  $N$  is `__l.size()`) and do no memory reallocation.

### **vector()** [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
std::vector< _Tp, _Alloc >::vector (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a = allocator_type()) [inline], [constexpr]
```

Builds a vector from a range.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |
| <code>__a</code>     | An allocator.      |

Create a vector consisting of copies of the elements from `[first,last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor  $N$  times (where  $N$  is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most  $2N$  calls to the copy constructor, and  $\log N$  memory reallocations.

### **~vector()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::~~vector () [inline], [constexpr], [noexcept]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 5.1011.3 Member Function Documentation

### **\_M\_allocate\_and\_copy()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _ForwardIterator>
pointer std::vector< _Tp, _Alloc >::_M_allocate_and_copy (
    size_type __n,
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [constexpr], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain  $n$  bytes of memory, and then copies `[first,last)` into it.

Referenced by `operator=()`, and `reserve()`.

### **\_M\_range\_check()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::_M_range_check (
    size_type __n) const [inline], [constexpr], [protected]
```

Safety check used only from `at()`.

Referenced by `std::vector< _Tp, polymorphic_allocator< _Tp > >::at()`, and `std::vector< _Tp, polymorphic_allocator< _Tp > >::at()`.

**assign()** [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::vector<_Tp, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last) [inline], [constexpr]
```

Assigns a range to a vector.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a vector with copies of the elements in the range [`__first`,`__last`).

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

**assign()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc >::assign (
    initializer_list<value_type> __l) [inline], [constexpr]
```

Assigns an initializer list to a vector.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>↵</code>   | An initializer_list. |
| <code>__↵</code> |                      |
| <code>↵</code>   |                      |
| <code>__↵</code> |                      |
| <code>l</code>   |                      |

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

**assign()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc >::assign (
    size_type __n,
    const value_type & __val) [inline], [constexpr]
```

Assigns a given value to a vector.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a vector with `__n` copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

**at()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::at (
    size_type __n) [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

**Parameters**

|       |                                                             |
|-------|-------------------------------------------------------------|
| $\_n$ | The index of the element for which data should be accessed. |
|-------|-------------------------------------------------------------|

**Returns**

Read/write reference to data.

**Exceptions**

|                          |                               |
|--------------------------|-------------------------------|
| <i>std::out_of_range</i> | If $\_n$ is an invalid index. |
|--------------------------|-------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws *out\_of\_range* if the check fails.

**at()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::at (
    size_type __n) const [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

**Parameters**

|       |                                                             |
|-------|-------------------------------------------------------------|
| $\_n$ | The index of the element for which data should be accessed. |
|-------|-------------------------------------------------------------|

**Returns**

Read-only (constant) reference to data.

**Exceptions**

|                          |                               |
|--------------------------|-------------------------------|
| <i>std::out_of_range</i> | If $\_n$ is an invalid index. |
|--------------------------|-------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws *out\_of\_range* if the check fails.

**back()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::back () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the vector.

**back()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector<_Tp, _Alloc >::back () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the last element of the vector.

```
begin() [1/2]
```

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc>::begin () const [inline], [nodiscard], [constexpr],
[noexcept]
```

**begin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::begin () [inline], [nodiscard], [constexpr], [noexcept]
Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.
```

Referenced by `std::vector<_Tp, polymorphic_allocator<_Tp>>::vector()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::crend()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::empty()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::erase()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::erase()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::front()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::front()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::insert()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::insert()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::insert()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::operator<=>()`, `operator=()`, `std::operator==()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::rend()` and `std::vector<_Tp, polymorphic_allocator<_Tp>>::rend()`.

**capacity()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::capacity () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Referenced by `operator=()`, and `reserve()`.

## cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::cbegin () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Referenced by `std::vector<_Tp, polymorphic_allocator<_Tp>>::erase()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::erase()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::insert()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::insert()`, and `std::vector<_Tp, polymorphic_allocator<_Tp>>::insert()`.

**cend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::cend () const [inline], [constexpr], [noexcept]
Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary
element order.
```

## clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::clear () [inline], [constexpr], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility. Referenced by [operator=\(\)](#).

### crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crbegin () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

### crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crend () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

### data()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
_Tp * std::vector< _Tp, _Alloc >::data () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a pointer such that `[data(), data() + size())` is a valid range. For a non-empty vector, `data() == &front()`.

### emplace()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename... _Args>
iterator std::vector< _Tp, _Alloc >::emplace (
    const_iterator __position,
    _Args &&... __args) [inline], [constexpr]
```

Inserts an object in vector before specified iterator.

#### Parameters

|                         |                                                |
|-------------------------|------------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the vector. |
| <code>__args</code>     | Arguments.                                     |

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.`

### empty()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::vector< _Tp, _Alloc >::empty () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns true if the vector is empty. (Thus `begin()` would equal `end()`.)

Referenced by [std::piecewise\\_constant\\_distribution< \\_RealType >::densities\(\)](#), [std::piecewise\\_linear\\_distribution< \\_RealType >::densities\(\)](#) and [std::discrete\\_distribution< \\_IntType >::probabilities\(\)](#).

**end()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc >::end () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

**end()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc >::end () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Referenced by [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp >>::vector\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp >>::back\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp >>::back\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp >>::cbegin\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp >>::empty\(\)](#), [\\_\\_gnu\\_parallel::multiway\\_merge\\_exact\\_splitting\(\)](#), [std::operator<=>\(\)](#), [operator=\(\)](#), [std::operator==\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp >>::rbegin\(\)](#), and [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp >>::rbegin\(\)](#).

**erase()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last) [inline], [constexpr]
```

Remove a range of elements.

**Parameters**

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

**Returns**

An iterator pointing to the element pointed to by `__last` prior to erasing (or `end()`).

This function will erase the elements in the range `[__first,__last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc >::erase (
    const_iterator __position) [inline], [constexpr]
```

Remove element at given position.

**Parameters**

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

**Returns**

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**front()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::front () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the vector.

**front()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::front () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the vector.

**get\_allocator()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::_Vector_base< _Tp, _Alloc >::get_allocator () const [inline], [constexpr],
[noexcept]
```

Get a copy of the memory allocation object.

References [get\\_allocator\(\)](#).

Referenced by [get\\_allocator\(\)](#).

**insert()** [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
iterator std::vector< _Tp, _Alloc >::insert (
    const_iterator __position,
    _InputIterator __first,
    _InputIterator __last) [inline], [constexpr]
```

Inserts a range into the vector.

**Parameters**

|                         |                                                |
|-------------------------|------------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the vector. |
| <code>__first</code>    | An input iterator.                             |
| <code>__last</code>     | An input iterator.                             |

**Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the range `[__first,__last)` into the vector before the location specified by `pos`.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**insert()** [2/5]

```
template<typename _Tp, typename _Alloc>
vector<_Tp, _Alloc >::iterator vector::insert (
    const_iterator __position,
    const value_type & __x) [constexpr]
```

Inserts given value into vector before specified iterator.

**Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__x</code>        | Data to be inserted.              |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

References [vector\(\)](#), [std::begin\(\)](#), [std::cbegin\(\)](#), [std::end\(\)](#), [insert\(\)](#), and [std::move\(\)](#).

Referenced by [insert\(\)](#).

**insert()** [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc >::insert (
    const_iterator __position,
    initializer_list<value_type> __l) [inline], [constexpr]
```

Inserts an initializer\_list into the vector.

**Parameters**

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__l</code>        | An initializer_list.         |

This function will insert copies of the data in the initializer\_list `l` into the vector before the location specified by `position`.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**insert()** [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc >::insert (
    const_iterator __position,
    size_type __n,
    const value_type & __x) [inline], [constexpr]
```

Inserts a number of copies of given data into the vector.

**Parameters**

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | A const_iterator into the vector.  |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |



**Returns**

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**insert()** [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
    const_iterator __position,
    value_type && __x) [inline], [constexpr]
```

Inserts given rvalue into vector before specified iterator.

**Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__x</code>        | Data to be inserted.              |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**max\_size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::max_size () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns the size() of the largest possible vector.

Referenced by [reserve\(\)](#).

**operator=()** [1/3]

```
template<typename _Tp, typename _Alloc>
vector< _Tp, _Alloc > & vector::operator= (
    const vector< _Tp, _Alloc > & __x) [constexpr]
```

Vector assignment operator.

**Parameters**

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

All the elements of `__x` are copied, but any unused capacity in `__x` will not be copied.

Whether the allocator is copied depends on the allocator traits.

References [vector\(\)](#), [std::\\_\\_addressof\(\)](#), [std::\\_Destroy\(\)](#), [\\_M\\_allocate\\_and\\_copy\(\)](#), [begin\(\)](#), [capacity\(\)](#), [clear\(\)](#), [std::end\(\)](#), [end\(\)](#), [std::size\(\)](#), and [size\(\)](#).

**operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector & std::vector<_Tp, _Alloc>::operator= (
    initializer_list<value_type> & __l) [inline], [constexpr]
```

Vector list assignment operator.

**Parameters**

|                                                                                   |                      |
|-----------------------------------------------------------------------------------|----------------------|
|  | An initializer_list. |
|  |                      |
|  |                      |
|  |                      |
|  |                      |

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector & std::vector<_Tp, _Alloc>::operator= (
    vector<_Tp, _Alloc> && __x) [inline], [constexpr], [noexcept]
```

Vector move assignment operator.

**Parameters**

|                                                                                     |                                                    |
|-------------------------------------------------------------------------------------|----------------------------------------------------|
|  | A vector of identical element and allocator types. |
|  |                                                    |

The contents of `__x` are moved into this vector (without copying, if the allocators permit it). Afterwards `__x` is a valid, but unspecified vector.

Whether the allocator is moved depends on the allocator traits.

**operator[]()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::operator[] (
    size_type __n) const [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

|                                                                                     |                                                             |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------|
|  | The index of the element for which data should be accessed. |
|  |                                                             |

**Returns**

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**operator[]()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::operator[] (
    size_type __n) [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

|                         |                                                             |
|-------------------------|-------------------------------------------------------------|
| <b><code>__n</code></b> | The index of the element for which data should be accessed. |
|-------------------------|-------------------------------------------------------------|

**Returns**

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out\_of\_range lookups are not defined. (For checked lookups see at().)

**pop\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::pop_back () [inline], [constexpr], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop\_back() is called.

**push\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::push_back (
    const value_type & __x) [inline], [constexpr]
```

Add data to the end of the vector.

**Parameters**

|                         |                   |
|-------------------------|-------------------|
| <b><code>__x</code></b> | Data to be added. |
|-------------------------|-------------------|

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

**rbegin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rbegin () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

**rbegin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc>::rbegin () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

**rend()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector<_Tp, _Alloc>::rend () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

**rend()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc>::rend () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

**reserve()**

```
template<typename _Tp, typename _Alloc>
void vector::reserve (
    size_type __n) [constexpr]
```

Attempt to preallocate enough memory for specified number of elements.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| $\leftrightarrow$ | Number of elements required. |
| $n$               |                              |

**Exceptions**

|                          |                                          |
|--------------------------|------------------------------------------|
| <i>std::length_error</i> | If $n$ exceeds <code>max_size()</code> . |
|--------------------------|------------------------------------------|

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

References [std::\\_Destroy\(\)](#), [\\_M\\_allocate\\_and\\_copy\(\)](#), [capacity\(\)](#), [max\\_size\(\)](#), and [std::size\(\)](#).

**resize()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::resize (
    size_type __new_size) [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain. |
|-------------------------|-----------------------------------------------|

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_shrink\\_and\\_double\(\)](#), and [\\_\\_gnu\\_parallel::multiway\\_merge\\_exact\\_splitting\(\)](#).

**resize()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (
    size_type __new_size,
    const value_type & __x) [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain.     |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

**shrink\_to\_fit()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::shrink_to_fit () [inline], [constexpr]
```

A non-binding request to reduce capacity() to size().

**size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::size () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns the number of elements in the vector.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_shrink\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_shrink\\_and\\_double\(\)](#), [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >::\\_\\_gnu\\_parallel::list\\_partition\(\)](#), [operator=\(\)](#), [std::operator==\(\)](#), [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >::resize\(\)](#), and [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >::resize\(\)](#).

**swap()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::swap (
    vector< _Tp, _Alloc > & __x) [inline], [constexpr], [noexcept]
```

Swaps data with another vector.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A vector of the same element and allocator types. |
|------------------|---------------------------------------------------|

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

The documentation for this class was generated from the following files:

- [stl\\_vector.h](#)
- [vector.tcc](#)

## 5.1012 std::vector< bool, \_Alloc > Class Template Reference

```
#include <vector>
```

Inheritance diagram for std::vector< bool, \_Alloc >:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Alloc` **allocator\_type**
- typedef `_Bit_const_iterator` **const\_iterator**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, vector >` **const\_iterator**
- typedef `const bool *` **const\_pointer**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `bool` **const\_reference**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Bit_iterator` **iterator**
- typedef `__gnu_cxx::__normal_iterator< pointer, vector >` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Base::pointer` **pointer**
- typedef `_Bit_reference` **reference**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `size_t` **size\_type**
- typedef `bool` **value\_type**
- typedef `bool` **value\_type**

## Public Member Functions

- `vector()`=default
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>  
constexpr vector(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `constexpr vector(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `constexpr vector(const allocator_type &__a) noexcept`
- `constexpr vector(const allocator_type &__a) noexcept`
- `constexpr vector(const vector &__x)`
- `constexpr vector(const vector &__x)`
- `constexpr vector(const vector &__x, const __type_identity_t< allocator_type > &__a)`
- `constexpr vector(const vector &__x, const __type_identity_t< allocator_type > &__a)`
- `constexpr vector(initializer_list< bool > __l, const allocator_type &__a=allocator_type())`
- `constexpr vector(initializer_list< value_type > __l, const allocator_type &__a=allocator_type())`
- `constexpr vector(size_type __n, const allocator_type &__a=allocator_type())`
- `constexpr vector(size_type __n, const allocator_type &__a=allocator_type())`
- `constexpr vector(size_type __n, const bool &__value, const allocator_type &__a=allocator_type())`
- `constexpr vector(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `vector(vector &&) noexcept=default`
- `vector(vector &&)=default`
- `constexpr vector(vector &&__rv, const __type_identity_t< allocator_type > &__m) noexcept(noexcept(vector(std::declval< vector && >(), std::declval< const allocator_type & >(), std::declval< typename _Alloc_traits::is_always_equal >())))`
- `constexpr vector(vector &&__x, const __type_identity_t< allocator_type > &__a) noexcept(_Bit_alloc_traits::↵  
_S_always_equal())`
- `constexpr ~vector() noexcept`
- `constexpr auto _M_emplace_aux(const_iterator __position, _Args &&... __args) -> iterator`
- `constexpr auto _M_emplace_aux(const_iterator __position, _Args &&... __args) -> iterator`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>  
constexpr void assign(_InputIterator __first, _InputIterator __last)`
- `constexpr void assign(_InputIterator __first, _InputIterator __last)`
- `constexpr void assign(initializer_list< bool > __l)`
- `constexpr void assign(initializer_list< value_type > __l)`
- `constexpr void assign(size_type __n, const bool &__x)`
- `constexpr void assign(size_type __n, const value_type &__val)`
- `constexpr reference at(size_type __n)`
- `constexpr reference at(size_type __n)`
- `constexpr const_reference at(size_type __n) const`
- `constexpr const_reference at(size_type __n) const`
- `constexpr reference back()`
- `constexpr const_reference back() const`
- `constexpr const_reference back() const noexcept`
- `constexpr reference back() noexcept`
- `constexpr const_iterator begin() const noexcept`
- `constexpr const_iterator begin() const noexcept`
- `constexpr iterator begin() noexcept`
- `constexpr iterator begin() noexcept`
- `constexpr size_type capacity() const noexcept`
- `constexpr size_type capacity() const noexcept`
- `constexpr const_iterator cbegin() const noexcept`
- `constexpr const_iterator cbegin() const noexcept`

- constexpr const\_iterator **cend** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr void **clear** () noexcept
- constexpr void **clear** () noexcept
- constexpr const\_reverse\_iterator **crbegin** () const noexcept
- constexpr const\_reverse\_iterator **crbegin** () const noexcept
- constexpr const\_reverse\_iterator **crend** () const noexcept
- constexpr const\_reverse\_iterator **crend** () const noexcept
- constexpr const bool \* **data** () const noexcept
- constexpr bool \* **data** () noexcept
- template<typename... \_Args>  
constexpr iterator **emplace** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- constexpr iterator **emplace** (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
constexpr reference **emplace\_back** (\_Args &&... \_\_args)
- constexpr reference **emplace\_back** (\_Args &&... \_\_args)
- constexpr bool **empty** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- constexpr iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- constexpr iterator **erase** (const\_iterator \_\_position)
- constexpr iterator **erase** (const\_iterator \_\_position)
- constexpr void **flip** () noexcept
- constexpr reference **front** ()
- constexpr const\_reference **front** () const
- constexpr const\_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- constexpr allocator\_type **get\_allocator** () const
- constexpr allocator\_type **get\_allocator** () const noexcept
- constexpr allocator\_type **get\_allocator** () const noexcept
- constexpr iterator **insert** (const\_iterator \_\_p, initializer\_list< bool > \_\_l)
- iterator **insert** (const\_iterator \_\_position)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>  
constexpr iterator **insert** (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- constexpr iterator **insert** (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- constexpr iterator **insert** (const\_iterator \_\_position, const bool &\_\_x)
- constexpr iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- constexpr iterator **insert** (const\_iterator \_\_position, initializer\_list< value\_type > \_\_l)
- constexpr iterator **insert** (const\_iterator \_\_position, size\_type \_\_n, const bool &\_\_x)
- constexpr iterator **insert** (const\_iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- constexpr iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- constexpr size\_type **max\_size** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- constexpr vector & **operator=** (const vector &\_\_x)
- constexpr vector & **operator=** (const vector &\_\_x)
- constexpr vector & **operator=** (initializer\_list< bool > \_\_l)
- constexpr vector & **operator=** (initializer\_list< value\_type > \_\_l)



- constexpr [vector](#) & [operator=](#) ([vector](#) && \_\_x) noexcept(\_Alloc\_traits::\_S\_nothrow\_move())
- constexpr [vector](#) & [operator=](#) ([vector](#) && \_\_x) noexcept(\_Bit\_alloc\_traits::\_S\_nothrow\_move())
- constexpr reference [operator\[\]](#) (size\_type \_\_n)
- constexpr const\_reference [operator\[\]](#) (size\_type \_\_n) const
- constexpr const\_reference [operator\[\]](#) (size\_type \_\_n) const noexcept
- constexpr reference [operator\[\]](#) (size\_type \_\_n) noexcept
- constexpr void [pop\\_back](#) ()
- constexpr void [pop\\_back](#) () noexcept
- constexpr void [push\\_back](#) (bool \_\_x)
- constexpr void [push\\_back](#) (const value\_type & \_\_x)
- constexpr void [push\\_back](#) (value\_type && \_\_x)
- constexpr [const\\_reverse\\_iterator](#) [rbegin](#) () const noexcept
- constexpr [const\\_reverse\\_iterator](#) [rbegin](#) () const noexcept
- constexpr [reverse\\_iterator](#) [rbegin](#) () noexcept
- constexpr [reverse\\_iterator](#) [rbegin](#) () noexcept
- constexpr [const\\_reverse\\_iterator](#) [rend](#) () const noexcept
- constexpr [const\\_reverse\\_iterator](#) [rend](#) () const noexcept
- constexpr [reverse\\_iterator](#) [rend](#) () noexcept
- constexpr [reverse\\_iterator](#) [rend](#) () noexcept
- constexpr void [reserve](#) (size\_type \_\_n)
- constexpr void [reserve](#) (size\_type \_\_n)
- constexpr void [resize](#) (size\_type \_\_new\_size)
- constexpr void [resize](#) (size\_type \_\_new\_size, bool \_\_x=bool())
- constexpr void [resize](#) (size\_type \_\_new\_size, const value\_type & \_\_x)
- constexpr void [shrink\\_to\\_fit](#) ()
- constexpr void [shrink\\_to\\_fit](#) ()
- constexpr size\_type [size](#) () const noexcept
- constexpr size\_type [size](#) () const noexcept
- constexpr void [swap](#) ([vector](#) & \_\_x) noexcept
- constexpr void [swap](#) ([vector](#) & \_\_x) noexcept

### Static Public Member Functions

- static constexpr void [swap](#) (reference \_\_x, reference \_\_y) noexcept

### Protected Types

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Alloc >::template rebind< \_Bit\_type >::other [\\_Bit\\_alloc\\_type](#)

### Protected Member Functions

- constexpr pointer [\\_M\\_allocate](#) (size\_t \_\_n)
- constexpr pointer [\\_M\\_allocate](#) (size\_t \_\_n)
- constexpr pointer [\\_M\\_allocate\\_and\\_copy](#) (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_ForwardIterator>  
constexpr void [\\_M\\_assign\\_aux](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- constexpr void [\\_M\\_assign\\_aux](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_InputIterator>  
constexpr void [\\_M\\_assign\\_aux](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- constexpr void [\\_M\\_assign\\_aux](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- constexpr void [\\_M\\_assign\\_dispatch](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- constexpr void [\\_M\\_assign\\_dispatch](#) (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)

- constexpr size\_type **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- constexpr size\_type **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- constexpr iterator **\_M\_copy\_aligned** (const\_iterator \_\_first, const\_iterator \_\_last, iterator \_\_result)
- constexpr void **\_M\_create\_storage** (size\_t \_\_n)
- constexpr void **\_M\_deallocate** ()
- constexpr void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- constexpr void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- constexpr void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- constexpr void **\_M\_default\_append** (size\_type \_\_n)
- constexpr void **\_M\_default\_initialize** (size\_type \_\_n)
- constexpr iterator **\_M\_emplace\_aux** (const\_iterator \_\_position, \_Args &&... \_\_args)
- constexpr iterator **\_M\_emplace\_aux** (const\_iterator \_\_position, value\_type &&\_\_v)
- constexpr iterator **\_M\_erase** (iterator \_\_first, iterator \_\_last)
- constexpr iterator **\_M\_erase** (iterator \_\_first, iterator \_\_last)
- constexpr iterator **\_M\_erase** (iterator \_\_pos)
- constexpr iterator **\_M\_erase** (iterator \_\_position)
- constexpr void **\_M\_erase\_at\_end** (iterator \_\_pos)
- constexpr void **\_M\_erase\_at\_end** (pointer \_\_pos) noexcept
- constexpr void **\_M\_fill\_append** (size\_type \_\_n, const value\_type &\_\_x)
- constexpr void **\_M\_fill\_assign** (size\_t \_\_n, bool \_\_x)
- constexpr void **\_M\_fill\_assign** (size\_type \_\_n, const value\_type &\_\_val)
- constexpr void **\_M\_fill\_initialize** (size\_type \_\_n, const value\_type &\_\_value)
- constexpr void **\_M\_fill\_insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- constexpr void **\_M\_fill\_insert** (iterator \_\_position, size\_type \_\_n, bool \_\_x)
- constexpr const \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () const noexcept
- constexpr \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () noexcept
- constexpr const \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- constexpr const \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- constexpr \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- constexpr \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- constexpr void **\_M\_initialize** (size\_type \_\_n)
- template<typename \_ForwardIterator>  
constexpr void **\_M\_initialize\_range** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_InputIterator>  
constexpr void **\_M\_initialize\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- constexpr void **\_M\_initialize\_value** (bool \_\_x) noexcept
- constexpr void **\_M\_insert\_aux** (iterator \_\_position, \_Arg &&\_\_arg)
- constexpr void **\_M\_insert\_aux** (iterator \_\_position, bool \_\_x)
- constexpr void **\_M\_insert\_dispatch** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- constexpr void **\_M\_insert\_dispatch** (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<typename \_InputIterator>  
constexpr void **\_M\_insert\_range** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator>  
constexpr void **\_M\_insert\_range** (iterator \_\_position, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- constexpr iterator **\_M\_insert\_rval** (const\_iterator \_\_position, value\_type &&\_\_v)
- constexpr void **\_M\_move\_data** (\_Bvector\_base &&\_\_x) noexcept
- constexpr void **\_M\_range\_check** (size\_type \_\_n) const
- constexpr void **\_M\_range\_check** (size\_type \_\_n) const
- constexpr void **\_M\_range\_initialize** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- constexpr void **\_M\_range\_initialize** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))

- constexpr void **\_M\_range\_initialize\_n** (\_Iterator \_\_first, \_Sentinel \_\_last, size\_type \_\_n)
- constexpr void **\_M\_range\_insert** (iterator \_\_pos, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- constexpr void **\_M\_range\_insert** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- constexpr void **\_M\_realloc\_append** (\_Args &&... \_\_args)
- constexpr void **\_M\_realloc\_insert** (iterator \_\_position, \_Args &&... \_\_args)
- constexpr void **\_M\_reallocate** (size\_type \_\_n)
- constexpr bool **\_M\_shrink\_to\_fit** ()
- constexpr bool **\_M\_shrink\_to\_fit** ()
- void **data** ()=delete

### Static Protected Member Functions

- static constexpr size\_type **\_S\_check\_init\_len** (size\_type \_\_n, const allocator\_type &\_\_a)
- static constexpr size\_type **\_S\_max\_size** (const \_Tp\_alloc\_type &\_\_a) noexcept
- static constexpr size\_t **\_S\_nword** (size\_t \_\_n)

### Protected Attributes

- \_Vector\_impl **\_M\_impl**
- \_Vector\_impl **\_M\_impl**

### Friends

- struct **std::hash< vector >**

### 5.1012.1 Detailed Description

**template<typename \_Alloc>**  
**class std::vector< bool, \_Alloc >**

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Since

C++98

### Template Parameters

|                     |                 |
|---------------------|-----------------|
| <code>_Alloc</code> | Allocator type. |
|---------------------|-----------------|

Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to `bool`. See DR96 for details.

See also

[vector](#) for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

### 5.1012.2 Constructor & Destructor Documentation

**vector()** [1/10]

`std::vector< bool, _Alloc >::vector ()` [default]

Creates a vector with no elements.

**vector()** [2/10]

```
std::vector< bool, _Alloc >::vector (
    const allocator_type & __a) [inline], [explicit], [constexpr], [noexcept]
```

Creates a vector with no elements.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

**vector()** [3/10]

```
std::vector< bool, _Alloc >::vector (
    size_type __n,
    const allocator_type & __a = allocator_type()) [inline], [explicit], [constexpr]
```

Creates a vector with default constructed elements.

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__n</code> | The number of elements to initially create. |
| <code>__a</code> | An allocator.                               |

This constructor fills the vector with `__n` default constructed elements.

**vector()** [4/10]

```
std::vector< bool, _Alloc >::vector (
    size_type __n,
    const value_type & __value,
    const allocator_type & __a = allocator_type()) [inline], [constexpr]
```

Creates a vector with copies of an exemplar element.

**Parameters**

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator.                               |

This constructor fills the vector with `__n` copies of `__value`.

**vector()** [5/10]

```
std::vector< bool, _Alloc >::vector (
    const vector< bool, _Alloc > & __x) [inline], [constexpr]
```

Vector copy constructor.

**Parameters**

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

All the elements of `__x` are copied, but any unused capacity in `__x` will not be copied (i.e. `capacity() == size()` in the new vector).

The newly-created vector uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

#### **vector()** [6/10]

```
std::vector< bool, _Alloc >::vector (
    vector< bool, _Alloc > && ) [default], [noexcept]
```

Vector move constructor.

The newly-created vector contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified vector.

#### **vector()** [7/10]

```
std::vector< bool, _Alloc >::vector (
    const vector< bool, _Alloc > & __x,
    const __type_identity_t< allocator_type > & __a) [inline], [constexpr]
```

Copy constructor with alternative allocator.

#### **vector()** [8/10]

```
std::vector< bool, _Alloc >::vector (
    vector< bool, _Alloc > && __rv,
    const __type_identity_t< allocator_type > & __m) [inline], [constexpr], [noexcept]
```

Move constructor with alternative allocator.

#### **vector()** [9/10]

```
std::vector< bool, _Alloc >::vector (
    initializer_list< value_type > __l,
    const allocator_type & __a = allocator_type()) [inline], [constexpr]
```

Builds a vector from an initializer list.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> . |
| <code>__a</code> | An allocator.                      |

Create a vector consisting of copies of the elements in the `initializer_list __l`.

This will call the element type's copy constructor `N` times (where `N` is `__l.size()`) and do no memory reallocation.

#### **vector()** [10/10]

```
std::vector< bool, _Alloc >::vector (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a = allocator_type()) [inline], [constexpr]
```

Builds a vector from a range.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |
| <code>__a</code>     | An allocator.      |

Create a vector consisting of copies of the elements from `[first,last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor  $N$  times (where  $N$  is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most  $2N$  calls to the copy constructor, and  $\log N$  memory reallocations.

`~vector()`

```
std::vector< bool, _Alloc >::~~vector () [inline], [constexpr], [noexcept]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 5.1012.3 Member Function Documentation

`_M_allocate_and_copy()`

```
pointer std::vector< bool, _Alloc >::_M_allocate_and_copy (
    size_type __n,
    _ForwardIterator __first,
    _ForwardIterator __last) [inline], [constexpr], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain  $n$  bytes of memory, and then copies `[first,last)` into it.

`_M_range_check()`

```
void std::vector< bool, _Alloc >::_M_range_check (
    size_type __n) const [inline], [constexpr], [protected]
```

Safety check used only from `at()`.

`assign()` [1/3]

```
void std::vector< bool, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last) [inline], [constexpr]
```

Assigns a range to a vector.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a vector with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

`assign()` [2/3]

```
void std::vector< bool, _Alloc >::assign (
    initializer_list< value_type > __l) [inline], [constexpr]
```

Assigns an initializer list to a vector.

## Parameters

|                      |                      |
|----------------------|----------------------|
| $\leftrightarrow$    | An initializer_list. |
| $\_ \leftrightarrow$ |                      |
| $\leftrightarrow$    |                      |
| $\_ \leftrightarrow$ |                      |
| $/$                  |                      |

This function fills a vector with copies of the elements in the initializer list  $\_ /$ .

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

**assign()** [3/3]

```
void std::vector< bool, _Alloc >::assign (
    size_type __n,
    const value_type & __val) [inline], [constexpr]
```

Assigns a given value to a vector.

## Parameters

|          |                                    |
|----------|------------------------------------|
| $\_ n$   | Number of elements to be assigned. |
| $\_ val$ | Value to be assigned.              |

This function fills a vector with  $\_ n$  copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

**at()** [1/2]

```
reference std::vector< bool, _Alloc >::at (
    size_type __n) [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| $\_ \leftrightarrow$ | The index of the element for which data should be accessed. |
| $\_ n$               |                                                             |

## Returns

Read/write reference to data.

## Exceptions

|                          |                                |
|--------------------------|--------------------------------|
| <i>std::out_of_range</i> | If $\_ n$ is an invalid index. |
|--------------------------|--------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws *out\_of\_range* if the check fails.

**at()** [2/2]

```
const_reference std::vector< bool, _Alloc >::at (
    size_type __n) const [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.



**Parameters**

|                                     |                                                             |
|-------------------------------------|-------------------------------------------------------------|
| <code>_↔</code><br><code>__n</code> | The index of the element for which data should be accessed. |
|-------------------------------------|-------------------------------------------------------------|

**Returns**

Read-only (constant) reference to data.

**Exceptions**

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

**back()** [1/2]

```
const_reference std::vector< bool, _Alloc >::back () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the vector.

**back()** [2/2]

```
reference std::vector< bool, _Alloc >::back () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the last element of the vector.

**begin()** [1/2]

```
const_iterator std::vector< bool, _Alloc >::begin () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

**begin()** [2/2]

```
iterator std::vector< bool, _Alloc >::begin () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

**capacity()**

```
size_type std::vector< bool, _Alloc >::capacity () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns the total number of elements that the vector can hold before needing to allocate more memory.

**cbegin()**

```
const_iterator std::vector< bool, _Alloc >::cbegin () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

**cend()**

```
const_iterator std::vector< bool, _Alloc >::cend () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

**clear()**

```
void std::vector< bool, _Alloc >::clear () [inline], [constexpr], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**crbegin()**

```
const_reverse_iterator std::vector< bool, _Alloc >::crbegin () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

**crend()**

```
const_reverse_iterator std::vector< bool, _Alloc >::crend () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

**data()**

```
bool * std::vector< bool, _Alloc >::data () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a pointer such that [data(), data() + size()) is a valid range. For a non-empty vector, data() == &front().

**emplace()**

```
iterator std::vector< bool, _Alloc >::emplace (
    const_iterator __position,
    _Args &&... __args) [inline], [constexpr]
```

Inserts an object in vector before specified iterator.

**Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__args</code>     | Arguments.                        |

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

**empty()**

```
bool std::vector< bool, _Alloc >::empty () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns true if the vector is empty. (Thus begin() would equal end().)

**end()** [1/2]

```
const_iterator std::vector< bool, _Alloc >::end () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

**end()** [2/2]

```
iterator std::vector< bool, _Alloc >::end () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

**erase()** [1/2]

```
iterator std::vector< bool, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last) [inline], [constexpr]
```

Remove a range of elements.

**Parameters**

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

**Returns**

An iterator pointing to the element pointed to by `__last` prior to erasing (or `end()`).

This function will erase the elements in the range `[__first,__last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/2]

```
iterator std::vector< bool, _Alloc >::erase (
    const_iterator __position) [inline], [constexpr]
```

Remove element at given position.

**Parameters**

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

**Returns**

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**front()** [1/2]

```
const_reference std::vector< bool, _Alloc >::front () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the vector.

**front()** [2/2]

```
reference std::vector< bool, _Alloc >::front () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the vector.

**get\_allocator()** [1/2]

```
() const
```

Get a copy of the memory allocation object.

**get\_allocator()** [2/2]

```
allocator_type std::_Vector_base< bool, _Alloc >::get_allocator () const [inline], [constexpr],  
[noexcept]
```

Get a copy of the memory allocation object.

**insert()** [1/5]

```
iterator std::vector< bool, _Alloc >::insert (  
    const_iterator __position,  
    _InputIterator __first,  
    _InputIterator __last) [inline], [constexpr]
```

Inserts a range into the vector.

**Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__first</code>    | An input iterator.                |
| <code>__last</code>     | An input iterator.                |

**Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the range [`__first`,`__last`) into the vector before the location specified by *pos*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**insert()** [2/5]

```
vector< bool, _Alloc >::iterator vector::insert (  
    const_iterator __position,  
    const value_type & __x) [constexpr]
```

Inserts given value into vector before specified iterator.

**Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__x</code>        | Data to be inserted.              |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**insert()** [3/5]

```
iterator std::vector< bool, _Alloc >::insert (
    const_iterator __position,
    initializer_list< value_type > __l) [inline], [constexpr]
```

Inserts an initializer\_list into the vector.

**Parameters**

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__l</code>        | An initializer_list.         |

This function will insert copies of the data in the initializer\_list *l* into the vector before the location specified by *position*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**insert()** [4/5]

```
iterator std::vector< bool, _Alloc >::insert (
    const_iterator __position,
    size_type __n,
    const value_type & __x) [inline], [constexpr]
```

Inserts a number of copies of given data into the vector.

**Parameters**

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | A const_iterator into the vector.  |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

**Returns**

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *position*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**insert()** [5/5]

```
iterator std::vector< bool, _Alloc >::insert (
    const_iterator __position,
    value_type && __x) [inline], [constexpr]
```

Inserts given rvalue into vector before specified iterator.

**Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__x</code>        | Data to be inserted.              |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**max\_size()**

```
size_type std::vector< bool, _Alloc >::max_size () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns the size() of the largest possible vector.

**operator=()** [1/3]

```
vector< bool, _Alloc > & vector::operator= (
    const vector< bool, _Alloc > & __x) [constexpr]
```

Vector assignment operator.

**Parameters**

|              |                                                    |
|--------------|----------------------------------------------------|
| $\leftarrow$ | A vector of identical element and allocator types. |
| $\_X$        |                                                    |

All the elements of  $\_x$  are copied, but any unused capacity in  $\_x$  will not be copied.  
Whether the allocator is copied depends on the allocator traits.

**operator=()** [2/3]

```
vector & std::vector< bool, _Alloc >::operator= (
    initializer_list< value_type > __l) [inline], [constexpr]
```

Vector list assignment operator.

**Parameters**

|                 |                      |
|-----------------|----------------------|
| $\leftarrow$    | An initializer_list. |
| $\_ \leftarrow$ |                      |
| $\leftarrow$    |                      |
| $\_ \leftarrow$ |                      |
| $/$             |                      |

This function fills a vector with copies of the elements in the initializer list  $\_/$ .

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
vector & std::vector< bool, _Alloc >::operator= (
    vector< bool, _Alloc > && __x) [inline], [constexpr], [noexcept]
```

Vector move assignment operator.

**Parameters**

|                 |                                                    |
|-----------------|----------------------------------------------------|
| $\_ \leftarrow$ | A vector of identical element and allocator types. |
| $\_X$           |                                                    |

The contents of  $\_x$  are moved into this vector (without copying, if the allocators permit it). Afterwards  $\_x$  is a valid, but unspecified vector.

Whether the allocator is moved depends on the allocator traits.

**operator[]()** [1/2]

```
const_reference std::vector< bool, _Alloc >::operator[] (
    size_type __n) const [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

|                                                                                   |                                                             |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------|
|  | The index of the element for which data should be accessed. |
| <code>__n</code>                                                                  |                                                             |

**Returns**

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**operator[]()** [2/2]

```
reference std::vector< bool, _Alloc >::operator[] (
    size_type __n) [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

|                                                                                   |                                                             |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------|
|  | The index of the element for which data should be accessed. |
| <code>__n</code>                                                                  |                                                             |

**Returns**

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**pop\_back()**

```
void std::vector< bool, _Alloc >::pop_back () [inline], [constexpr], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

**push\_back()**

```
void std::vector< bool, _Alloc >::push_back (
    const value_type & __x) [inline], [constexpr]
```

Add data to the end of the vector.

**Parameters**

|                                                                                     |                   |
|-------------------------------------------------------------------------------------|-------------------|
|  | Data to be added. |
| <code>__x</code>                                                                    |                   |

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

**rbegin()** [1/2]

```
const_reverse_iterator std::vector< bool, _Alloc >::rbegin () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

**rbegin()** [2/2]

```
reverse_iterator std::vector< bool, _Alloc >::rbegin () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

**rend()** [1/2]

```
const_reverse_iterator std::vector< bool, _Alloc >::rend () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

**rend()** [2/2]

```
reverse_iterator std::vector< bool, _Alloc >::rend () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

**reserve()**

```
void vector::reserve (
    size_type __n) [constexpr]
```

Attempt to preallocate enough memory for specified number of elements.

**Parameters**

|                                       |                              |
|---------------------------------------|------------------------------|
| $\leftrightarrow$<br><code>__n</code> | Number of elements required. |
|---------------------------------------|------------------------------|

**Exceptions**

|                                |                                                     |
|--------------------------------|-----------------------------------------------------|
| <code>std::length_error</code> | If <code>n</code> exceeds <code>max_size()</code> . |
|--------------------------------|-----------------------------------------------------|

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

**resize()** [1/2]

```
void std::vector< bool, _Alloc >::resize (
    size_type __new_size) [inline], [constexpr]
```

Resizes the vector to the specified number of elements.



**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain. |
|-------------------------|-----------------------------------------------|

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

**resize()** [2/2]

```
void std::vector< bool, _Alloc >::resize (
    size_type __new_size,
    const value_type & __x) [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain.     |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

**shrink\_to\_fit()**

```
void std::vector< bool, _Alloc >::shrink_to_fit () [inline], [constexpr]
```

A non-binding request to reduce capacity() to size().

**size()**

```
size_type std::vector< bool, _Alloc >::size () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns the number of elements in the vector.

**swap()**

```
void std::vector< bool, _Alloc >::swap (
    vector< bool, _Alloc > & __x) [inline], [constexpr], [noexcept]
```

Swaps data with another vector.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A vector of the same element and allocator types. |
| <code>__y</code> |                                                   |

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

The documentation for this class was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

**5.1013 std::ranges::view\_interface< \_Derived > Class Template Reference**

```
#include <ranges_util.h>
```

Inheritance diagram for std::ranges::view\_interface< \_Derived >:

**Public Member Functions**

- constexpr decltype(auto) **back** ()
- constexpr decltype(auto) **back** () const
- constexpr auto **data** () const noexcept(noexcept(ranges::begin(\_M\_derived())))
- constexpr auto **data** () noexcept(noexcept(ranges::begin(\_M\_derived())))
- constexpr bool **empty** () const noexcept(noexcept(\_S\_empty(\_M\_derived())))
- constexpr bool **empty** () const noexcept(noexcept(ranges::size(\_M\_derived())==0))
- constexpr bool **empty** () noexcept(noexcept(\_S\_empty(\_M\_derived())))
- constexpr bool **empty** () noexcept(noexcept(ranges::size(\_M\_derived())==0))
- constexpr decltype(auto) **front** ()
- constexpr decltype(auto) **front** () const
- constexpr **operator bool** () const noexcept(noexcept(ranges::empty(\_M\_derived())))
- constexpr **operator bool** () noexcept(noexcept(ranges::empty(\_M\_derived())))
- template<random\_access\_range \_Range = \_Derived>  
constexpr decltype(auto) **operator[]** (range\_difference\_t< \_Range > \_\_n)
- template<random\_access\_range \_Range = const \_Derived>  
constexpr decltype(auto) **operator[]** (range\_difference\_t< \_Range > \_\_n) const
- constexpr auto **size** () const noexcept(noexcept(\_S\_size(\_M\_derived())))
- constexpr auto **size** () noexcept(noexcept(\_S\_size(\_M\_derived())))

**5.1013.1 Detailed Description**

```
template<typename _Derived>
```

```
requires is_class_v<_Derived> && same_as<_Derived, remove_cv_t<_Derived>>
```

```
class std::ranges::view_interface< _Derived >
```

The ranges::view\_interface class template.

The documentation for this class was generated from the following file:

- [ranges\\_util.h](#)

## 5.1014 `std::wbuffer_convert<_Codecvt, _Elem, _Tr>` Class Template Reference

```
#include <locale_conv.h>
```

Inheritance diagram for `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`:



### Public Types

- typedef `_Codecvt::state_type` **state\_type**
- typedef `_CharT` **char\_type**
- typedef `_Traits` **traits\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::pos_type` **pos\_type**
- typedef `traits_type::off_type` **off\_type**
- typedef `basic_streambuf< char_type, traits_type >` **\_\_streambuf\_type**

### Public Member Functions

- **wbuffer\_convert** ()
- **wbuffer\_convert** (const **wbuffer\_convert** &)=delete
- **wbuffer\_convert** (**streambuf** \* \_\_bytebuf, `_Codecvt` \* \_\_pcvt=new `_Codecvt`, `state_type` \_\_state=state\_type())
- **locale** **getloc** () const
- **streamsize** **in\_avail** ()
- **wbuffer\_convert** & **operator=** (const **wbuffer\_convert** &)=delete
- **locale** **pubimbue** (const **locale** & \_\_loc)
- **streambuf** \* **rdbuf** () const noexcept
- **streambuf** \* **rdbuf** (**streambuf** \* \_\_bytebuf) noexcept
- `int_type` **sbumpc** ()
- `int_type` **sgetc** ()
- **streamsize** **sgetn** (`char_type` \* \_\_s, **streamsize** \_\_n)
- `int_type` **snextc** ()
- `int_type` **sputbackc** (`char_type` \_\_c)
- `int_type` **sputc** (`char_type` \_\_c)

- [streamsize sputn](#) (const [char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [state\\_type state](#) () const noexcept
- [int\\_type sungetc](#) ()
- [basic\\_streambuf](#) \* [pubsetbuf](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [pos\\_type](#) [pubseekoff](#) ([off\\_type](#) \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [pos\\_type](#) [pubseekpos](#) ([pos\\_type](#) \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [int](#) [pubsync](#) ()

### Protected Member Functions

- void [\\_\\_safe\\_gbump](#) ([streamsize](#) \_\_n)
- void [\\_\\_safe\\_pbump](#) ([streamsize](#) \_\_n)
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const [locale](#) & \_\_loc)
- [\\_Wide\\_streambuf::int\\_type](#) [overflow](#) (typename [\\_Wide\\_streambuf::int\\_type](#) \_\_out)
- virtual [int\\_type](#) [pbackfail](#) ([int\\_type](#) \_\_c=traits\_type::eof())
- void [pbump](#) (int \_\_n)
- virtual [pos\\_type](#) [seekoff](#) ([off\\_type](#), [ios\\_base::seekdir](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual [pos\\_type](#) [seekpos](#) ([pos\\_type](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual [basic\\_streambuf](#)< [char\\_type](#), [\\_Traits](#) > \* [setbuf](#) ([char\\_type](#) \*, [streamsize](#))
- void [setg](#) ([char\\_type](#) \* \_\_gbeg, [char\\_type](#) \* \_\_gnext, [char\\_type](#) \* \_\_gend)
- void [setp](#) ([char\\_type](#) \* \_\_pbeg, [char\\_type](#) \* \_\_pend)
- virtual [streamsize](#) [showmanyc](#) ()
- void [swap](#) ([basic\\_streambuf](#) & \_\_sb)
- [int](#) [sync](#) ()
- virtual [int\\_type](#) [uflow](#) ()
- [\\_Wide\\_streambuf::int\\_type](#) [underflow](#) ()
- virtual [streamsize](#) [xsgetn](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- virtual [streamsize](#) [xsputn](#) (const [char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [streamsize](#) [xspn](#) (const typename [\\_Wide\\_streambuf::char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [char\\_type](#) \* [eback](#) () const
- [char\\_type](#) \* [gptr](#) () const
- [char\\_type](#) \* [egptr](#) () const
- [char\\_type](#) \* [pbase](#) () const
- [char\\_type](#) \* [pptr](#) () const
- [char\\_type](#) \* [epptr](#) () const

### Protected Attributes

- [locale](#) [\\_M\\_buf\\_locale](#)
- [char\\_type](#) \* [\\_M\\_in\\_beg](#)
- [char\\_type](#) \* [\\_M\\_in\\_cur](#)
- [char\\_type](#) \* [\\_M\\_in\\_end](#)
- [char\\_type](#) \* [\\_M\\_out\\_beg](#)
- [char\\_type](#) \* [\\_M\\_out\\_cur](#)
- [char\\_type](#) \* [\\_M\\_out\\_end](#)

### 5.1014.1 Detailed Description

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
class std::wbuffer_convert< _Codecvt, _Elem, _Tr >
```

Buffer conversions.

### 5.1014.2 Member Typedef Documentation

#### **\_\_streambuf\_type**

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _CharT, _Traits >::__↵
streambuf_type [inherited]
This is a non-standard type.
```

#### **char\_type**

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_streambuf< _CharT, _Traits >::char_type [inherited]
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the
template parameters, which are specific to the implementation.
```

#### **int\_type**

```
template<typename _CharT, typename _Traits>
typedef traits_type::int_type std::basic_streambuf< _CharT, _Traits >::int_type [inherited]
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the
template parameters, which are specific to the implementation.
```

#### **off\_type**

```
template<typename _CharT, typename _Traits>
typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type [inherited]
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the
template parameters, which are specific to the implementation.
```

#### **pos\_type**

```
template<typename _CharT, typename _Traits>
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type [inherited]
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the
template parameters, which are specific to the implementation.
```

#### **traits\_type**

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type [inherited]
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the
template parameters, which are specific to the implementation.
```

### 5.1014.3 Constructor & Destructor Documentation

#### **wbuffer\_convert()** [1/2]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert () [inline]
```

Default constructor.

References [wbuffer\\_convert\(\)](#).

Referenced by [wbuffer\\_convert\(\)](#).

## wbuffer\_convert() [2/2]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert (
    streambuf * __bytebuf,
    _Codecvt * __pcvt = new _Codecvt,
    state_type __state = state_type()) [inline], [explicit]
```

Constructor.

### Parameters

|                        |                                    |
|------------------------|------------------------------------|
| <code>__bytebuf</code> | The underlying byte stream buffer. |
| <code>__pcvt</code>    | The facet to use for conversions.  |
| <code>__state</code>   | Initial conversion state.          |

Takes ownership of `__pcvt` and will delete it in the destructor.

References [std::basic\\_streambuf<\\_CharT, \\_Traits >::setg\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits >::setp\(\)](#).

## 5.1014.4 Member Function Documentation

### eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits >::underflow\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits >::xsgetn\(\)](#).

### egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits >::showmanyc\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits >::underflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::underflow\(\)](#), [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr >::underflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits >::xsgetn\(\)](#), and [xsgetn\(\)](#).

### epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::xsputn()`, and `xsputn()`.

### gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
    int __n) [inline], [protected], [inherited]
```

Moving the read position.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

### getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline], [inherited]
```

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

### gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::imbue\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::showmanyc\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::underflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::underflow\(\)](#), [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::underflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#), and [xsgetn\(\)](#).

### imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf<_CharT, _Traits>::imbue (
    const locale & __loc) [inline], [protected], [virtual], [inherited]
```

Changes translations.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<wchar\\_t>](#), and [std::basic\\_filebuf<wchar\\_t>](#).

### in\_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

### overflow()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert<_Codecvt, _Elem, _Tr>::overflow (
    typename _Wide_streambuf::int_type __c) [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|



**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `std::basic_streambuf<_CharT, _Traits>::sputc()`.

**pbackfail()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::pbackfail (
    int_type __c = traits_type::eof()) [inline], [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_stringbuf<char>`, `std::basic_stringbuf<char>`, `std::basic_stringbuf<wchar_t>`, and `std::basic_stringbuf<wchar_t>`.

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::sync()`, and `std::basic_filebuf<_CharT, _Traits>::xspn()`.

**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
    int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#).

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived `imbue(__loc)`.

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

**pubsetbuf()**

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

**pubsync()**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]
```

Calls virtual sync function.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**sbumpc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`.

**seekoff()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_filebuf< wchar_t >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_stringbuf< char >`, `std::basic_stringbuf< char >`, `std::basic_stringbuf< wchar_t >`, and `std::basic_stringbuf< wchar_t >`.

**seekpos()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_filebuf< wchar_t >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_stringbuf< char >`, `std::basic_stringbuf< char >`, `std::basic_stringbuf< wchar_t >`, and `std::basic_stringbuf< wchar_t >`.

**setbuf()**

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
    char_type * ,
    streamsize ) [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

**Note**

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_stringbuf< char >`, and `std::basic_stringbuf< char >`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >::wbuffer\\_convert\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
    char_type * __pbeg,
    char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >::wbuffer\\_convert\(\)](#).

**sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::getline\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), and [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#).

**sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
    char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for xsgetn.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns xsgetn(\_\_s,\_\_n). The effect is to fill \_\_s[0] through \_\_s[\_\_n-1] with characters from the input sequence, if possible.

**showmanyc()**

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc () [inline], [protected],
[virtual], [inherited]
Investigating the data available.
```

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to underflow() will not return traits::eof() until at least that number of characters have been supplied. If showmanyc() returns -1, then calls to underflow() or uflow() will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return eof() but that they will return immediately.*

The standard adds that *the morphemes of showmanyc are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_filebuf< wchar_t >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_stringbuf< char >`, `std::basic_stringbuf< char_type, traits_type >`, `std::basic_stringbuf< wchar_t >`, and `std::basic_stringbuf< wchar_t >`.

**snextc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls sbumpc(), and if that function returns traits::eof(), so does this function. Otherwise, sgetc().

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< char_type, traits_type >::ignore()`, `std::basic_istream< char_type, traits_type >::operator>>()`, and `std::basic_istream< char_type, traits_type >::putback()`.

**sputbackc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
    char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::putc (
    char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
    const char_type * __s,
    streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

---

One of two public output functions.

Returns `xspn(____s, ____n)`. The effect is to write `____s[0]` through `____s[____n-1]` to the output sequence, if possible.

### **state()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
state_type std::wbuffer_convert<_Codecvt, _Elem, _Tr>::state () const [inline], [noexcept]
```

The conversion state following the last conversion.

### **sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

#### **Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream<char_type, traits_type>::sentry::sentry()`.

### **sync()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
int std::wbuffer_convert<_Codecvt, _Elem, _Tr>::sync () [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

#### **Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

#### **Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

### **uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

#### **Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Referenced by `xsgn()`.



**underflow()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow () [inline],
[protected], [virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `std::basic_streambuf< _CharT, _Traits >::egptr()`, and `std::basic_streambuf< _CharT, _Traits >::gptr()`.

**xsgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
    char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_filebuf< wchar_t >`.

References `egptr()`, `gptr()`, `std::min()`, and `uflow()`.

**xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
    const char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_filebuf<wchar_t>`.

References `epptr()`, `std::min()`, `overflow()`, and `pptr()`.

## 5.1014.5 Member Data Documentation

**`_M_buf_locale`**

```
template<typename _CharT, typename _Traits>
```

```
locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

**`_M_in_beg`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_beg [protected], [inherited]
```

Start of get area.

**`_M_in_cur`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_cur [protected], [inherited]
```

Current read area.

**`_M_in_end`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_end [protected], [inherited]
```

End of get area.

**`_M_out_beg`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_beg [protected], [inherited]
```

Start of put area.

**`_M_out_cur`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_cur [protected], [inherited]
```

Current put area.

## **\_M\_out\_end**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

The documentation for this class was generated from the following file:

- [locale\\_conv.h](#)

## **5.1015 std::weak\_ptr< \_Tp > Class Template Reference**

```
#include <memory>
```

### **Public Types**

- using **element\_type**

### **Public Member Functions**

- template<typename \_Yp, typename = \_Constructible<const shared\_ptr<\_Yp>&>>  
**weak\_ptr** (const [shared\\_ptr](#)< \_Yp > &\_\_r) noexcept
- **weak\_ptr** (const [weak\\_ptr](#) &) noexcept=default
- template<typename \_Yp, typename = \_Constructible<const weak\_ptr<\_Yp>&>>  
**weak\_ptr** (const [weak\\_ptr](#)< \_Yp > &\_\_r) noexcept
- **weak\_ptr** ([weak\\_ptr](#) &&) noexcept=default
- template<typename \_Yp, typename = \_Constructible<weak\_ptr<\_Yp>>>  
**weak\_ptr** ([weak\\_ptr](#)< \_Yp > &&\_\_r) noexcept
- bool **expired** () const noexcept
- [shared\\_ptr](#)< \_Tp > **lock** () const noexcept
- template<typename \_Yp>  
\_Assignable< const [shared\\_ptr](#)< \_Yp > & > **operator=** (const [shared\\_ptr](#)< \_Yp > &\_\_r) noexcept
- [weak\\_ptr](#) & **operator=** (const [weak\\_ptr](#) &\_\_r) noexcept=default
- template<typename \_Yp>  
\_Assignable< const [weak\\_ptr](#)< \_Yp > & > **operator=** (const [weak\\_ptr](#)< \_Yp > &\_\_r) noexcept
- [weak\\_ptr](#) & **operator=** ([weak\\_ptr](#) &&\_\_r) noexcept=default
- template<typename \_Yp>  
\_Assignable< [weak\\_ptr](#)< \_Yp > > **operator=** ([weak\\_ptr](#)< \_Yp > &&\_\_r) noexcept
- template<typename \_Tp1>  
bool **owner\_before** (const \_\_shared\_ptr< \_Tp1, \_Lp > &\_\_rhs) const noexcept
- template<typename \_Tp1>  
bool **owner\_before** (const \_\_weak\_ptr< \_Tp1, \_Lp > &\_\_rhs) const noexcept
- void **reset** () noexcept
- void **swap** (\_\_weak\_ptr &\_\_s) noexcept
- long **use\_count** () const noexcept

### **Related Symbols**

(Note that these are not member symbols.)

- template<typename \_Tp>  
void **swap** ([weak\\_ptr](#)< \_Tp > &\_\_a, [weak\\_ptr](#)< \_Tp > &\_\_b) noexcept

### 5.1015.1 Detailed Description

**template<typename \_Tp>**  
**class std::weak\_ptr<\_Tp>**

A non-owning observer for a pointer owned by a shared\_ptr.

Since

C++11

A weak\_ptr provides a safe alternative to a raw pointer when you want a non-owning reference to an object that is managed by a shared\_ptr.

Unlike a raw pointer, a weak\_ptr can be converted to a new shared\_ptr that shares ownership with every other shared\_ptr that already owns the pointer. In other words you can upgrade from a non-owning "weak" reference to an owning shared\_ptr, without having access to any of the existing shared\_ptr objects.

Also unlike a raw pointer, a weak\_ptr does not become "dangling" after the object it points to has been destroyed. Instead, a weak\_ptr becomes *expired* and can no longer be converted to a shared\_ptr that owns the freed pointer, so you cannot accidentally access the pointed-to object after it has been destroyed.

The documentation for this class was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 5.1016 std::weibull\_distribution<\_RealType> Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **weibull\_distribution** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- **weibull\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool `operator==` (const `weibull_distribution` &\_\_d1, const `weibull_distribution` &\_\_d2)

### 5.1016.1 Detailed Description

```
template<typename _RealType = double>
class std::weibull_distribution< _RealType >
```

A `weibull_distribution` random number distribution.  
The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^{\alpha}\right)$$

Since

C++11

### 5.1016.2 Member Typedef Documentation

#### `result_type`

```
template<typename _RealType = double>
typedef _RealType std::weibull_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 5.1016.3 Member Function Documentation

#### `a()`

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::a () const [inline]
```

Return the *a* parameter of the distribution.

#### `b()`

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::b () const [inline]
```

Return the *b* parameter of the distribution.

#### `max()`

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### `min()`

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### `operator()()`

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
```

```
result_type std::weibull_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

#### param() [1/2]

```
template<typename _RealType = double>
param_type std::weibull_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

#### param() [2/2]

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::param (
    const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

#### reset()

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

### 5.1016.4 Friends And Related Symbol Documentation

#### operator==

```
template<typename _RealType = double>
bool operator== (
    const weibull_distribution< _RealType > & __d1,
    const weibull_distribution< _RealType > & __d2) [friend]
```

Return true if two Weibull distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.1017 std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc > Class Template Reference

```
#include <locale_conv.h>
```

#### Public Types

- typedef [basic\\_string](#)< char, [char\\_traits](#)< char >, \_Byte\_alloc > **byte\_string**
- typedef [wide\\_string::traits\\_type::int\\_type](#) **int\_type**
- typedef [\\_Codecvt::state\\_type](#) **state\_type**
- typedef [basic\\_string](#)< \_Elem, [char\\_traits](#)< \_Elem >, \_Wide\_alloc > **wide\_string**

## Public Member Functions

- [wstring\\_convert](#) ()
  - [wstring\\_convert](#) (\_Codecvt \* \_\_pcvt)
  - [wstring\\_convert](#) (\_Codecvt \* \_\_pcvt, state\_type \_\_state)
  - [wstring\\_convert](#) (const [byte\\_string](#) & \_\_byte\_err, const [wide\\_string](#) & \_\_wide\_err=[wide\\_string](#)())
  - [wstring\\_convert](#) (const [wstring\\_convert](#) &)=delete
  - [size\\_t converted](#) () const noexcept
  - [wstring\\_convert](#) & **operator=** (const [wstring\\_convert](#) &)=delete
  - [state\\_type state](#) () const
- 
- [wide\\_string from\\_bytes](#) (char \_\_byte)
  - [wide\\_string from\\_bytes](#) (const char \* \_\_ptr)
  - [wide\\_string from\\_bytes](#) (const [byte\\_string](#) & \_\_str)
  - [wide\\_string from\\_bytes](#) (const char \* \_\_first, const char \* \_\_last)
- 
- [byte\\_string to\\_bytes](#) (\_Elem \_\_wchar)
  - [byte\\_string to\\_bytes](#) (const \_Elem \* \_\_ptr)
  - [byte\\_string to\\_bytes](#) (const [wide\\_string](#) & \_\_wstr)
  - [byte\\_string to\\_bytes](#) (const \_Elem \* \_\_first, const \_Elem \* \_\_last)

### 5.1017.1 Detailed Description

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>, type-
name _Byte_alloc = allocator<char>>
class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >
```

String conversions.

### 5.1017.2 Constructor & Destructor Documentation

#### [wstring\\_convert](#)() [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert () [inline]
Default constructor.
```

#### [wstring\\_convert](#)() [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
    _Codecvt * __pcvt) [inline], [explicit]
```

Constructor.

#### Parameters

|                     |                                   |
|---------------------|-----------------------------------|
| <code>__pcvt</code> | The facet to use for conversions. |
|---------------------|-----------------------------------|

Takes ownership of `__pcvt` and will delete it in the destructor.

**wstring\_convert()** [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
    _Codecvt * __pcvt,
    state_type __state) [inline]
```

Construct with an initial conversion state.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__pcvt</code>  | The facet to use for conversions. |
| <code>__state</code> | Initial conversion state.         |

Takes ownership of `__pcvt` and will delete it in the destructor. The object's conversion state will persist between conversions.

**wstring\_convert()** [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
    const byte_string & __byte_err,
    const wide_string & __wide_err = wide_string()) [inline], [explicit]
```

Construct with error strings.

**Parameters**

|                         |                                                |
|-------------------------|------------------------------------------------|
| <code>__byte_err</code> | A string to return on failed conversions.      |
| <code>__wide_err</code> | A wide string to return on failed conversions. |

**5.1017.3 Member Function Documentation****converted()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
size_t std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::converted () const
[inline], [noexcept]
```

The number of elements successfully converted in the last conversion.

**from\_bytes()** [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
    char __byte) [inline]
```

Convert from bytes.

References [from\\_bytes\(\)](#).

Referenced by [from\\_bytes\(\)](#), [from\\_bytes\(\)](#), and [from\\_bytes\(\)](#).



**from\_bytes()** [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
    const byte_string & __str) [inline]
```

Convert from bytes.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::data\(\)](#), [from\\_bytes\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).

**from\_bytes()** [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
    const char * __first,
    const char * __last) [inline]
```

Convert from bytes.

**from\_bytes()** [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
    const char * __ptr) [inline]
```

Convert from bytes.

References [from\\_bytes\(\)](#).

**state()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
state_type std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::state () const
[inline]
```

The final conversion state of the last conversion.

**to\_bytes()** [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
    _Elem __wchar) [inline]
```

Convert to bytes.

References [to\\_bytes\(\)](#).

Referenced by [to\\_bytes\(\)](#), [to\\_bytes\(\)](#), and [to\\_bytes\(\)](#).

**to\_bytes()** [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
    const _Elem * __first,
    const _Elem * __last) [inline]
```

Convert to bytes.

**to\_bytes()** [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
    const _Elem * __ptr) [inline]
```

Convert to bytes.

References [std::basic\\_string< \\_Elem, char\\_traits< \\_Elem >, \\_Wide\\_alloc >::length\(\)](#), and [to\\_bytes\(\)](#).

**to\_bytes()** [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
    const wide_string & __wstr) [inline]
```

Convert to bytes.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::data\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::size\(\)](#), and [to\\_bytes\(\)](#).

The documentation for this class was generated from the following file:

- [locale\\_conv.h](#)

## 6 File Documentation

### 6.1 algorithm File Reference

#### Macros

- `#define __glibcxx_want_algorithm_default_value_type`
- `#define __glibcxx_want_clamp`
- `#define __glibcxx_want_constexpr_algorithms`
- `#define __glibcxx_want_freestanding_algorithm`
- `#define __glibcxx_want_parallel_algorithm`
- `#define __glibcxx_want_ranges`
- `#define __glibcxx_want_ranges_contains`
- `#define __glibcxx_want_ranges_find_last`
- `#define __glibcxx_want_ranges_fold`
- `#define __glibcxx_want_ranges_starts_ends_with`
- `#define __glibcxx_want_robust_nonmodifying_seq_ops`
- `#define __glibcxx_want_sample`
- `#define __glibcxx_want_shift`
- `#define _GLIBCXX_ALGORITHM`

#### 6.1.1 Detailed Description

This is a Standard C++ Library header.

### 6.2 experimental/algorithm File Reference

#### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

## Macros

- `#define __cpp_lib_experimental_sample`
- `#define _GLIBCXX_EXPERIMENTAL_ALGORITHM`

## Functions

- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance>`  
`_SampleIterator std::experimental::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>`  
`_SampleIterator std::experimental::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`_ForwardIterator std::experimental::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`
- `template<typename _RandomAccessIterator>`  
`void std::experimental::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`

### 6.2.1 Detailed Description

This is a TS C++ Library header.

## 6.3 ext/algorithm File Reference

### Namespaces

- namespace `__gnu_cxx`

### Macros

- `#define _EXT_ALGORITHM`

### Functions

- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`std::pair< _InputIterator, _OutputIterator > gnu_cxx::__copy_n (_InputIterator __first, _Size __count, _OutputIterator __result, std::input_iterator_tag)`
- `template<typename _RAIterator, typename _Size, typename _OutputIterator>`  
`std::pair< _RAIterator, _OutputIterator > gnu_cxx::__copy_n (_RAIterator __first, _Size __count, _OutputIterator __result, std::random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`int gnu_cxx::__lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `int gnu_cxx::__lexicographical_compare_3way (const char *__first1, const char *__last1, const char *__first2, const char *__last2)`
- `int gnu_cxx::__lexicographical_compare_3way (const unsigned char *__first1, const unsigned char *__last1, const unsigned char *__first2, const unsigned char *__last2)`
- `template<typename _Tp>`  
`const _Tp & gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare>`  
`const _Tp & gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance>`  
`_RandomAccessIterator gnu_cxx::__random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`

- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance>`  
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, ↵`  
`RandomAccessIterator __out, const _Distance __n)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _Output↵`  
`Iterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size>`  
`void __gnu_cxx::count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size>`  
`void __gnu_cxx::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _RAIter>`  
`constexpr bool __gnu_cxx::is_heap (_RAIter, _RAIter)`
- `template<typename _Filter>`  
`constexpr bool __gnu_cxx::is_sorted (_Filter, _Filter)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 ↵`  
`__first2, _InputIterator2 __last2)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`  
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↵`  
`AccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↵`  
`AccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>`  
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator ↵`  
`__out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>`  
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator ↵`  
`__out, const _Distance __n, _RandomNumberGenerator &__rand)`

### 6.3.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.4 parallel/algorithm File Reference

### Macros

- `#define _PARALLEL_ALGORITHM`

### 6.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.5 any File Reference

### Macros

- `#define __glibcxx_want_any`
- `#define _GLIBCXX_ANY`

### 6.5.1 Detailed Description

This is a Standard C++ Library header.

## 6.6 experimental/any File Reference

### Classes

- class [std::experimental::fundamentals\\_v1::any](#)
- class [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_any`
- `#define _GLIBCXX_EXPERIMENTAL_ANY`

### Functions

- `template<typename _ValueType>`  
`_ValueType std::experimental::any_cast (const any &__any)`
- `void std::experimental::swap (any &__x, any &__y) noexcept`
- `template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>`  
`_ValueType std::experimental::any_cast (any &&__any)`
- `template<typename _ValueType, typename enable_if< is_move_constructible< _ValueType >::value &&!is_lvalue_reference< _ValueType >::value, bool >::type = false>`  
`_ValueType std::experimental::any_cast (any &&__any)`
- `template<typename _ValueType>`  
`_ValueType std::experimental::any_cast (any &__any)`
- `template<typename _ValueType>`  
`_ValueType * std::experimental::any_cast (any *__any) noexcept`
- `template<typename _ValueType>`  
`const _ValueType * std::experimental::any_cast (const any *__any) noexcept`

#### 6.6.1 Detailed Description

This is a TS C++ Library header.

## 6.7 array File Reference

### Classes

- struct [std::array<\\_Tp, \\_Nm >](#)
- struct [std::tuple\\_element<\\_Ind, array<\\_Tp, \\_Nm > >](#)
- struct [std::tuple\\_size< array<\\_Tp, \\_Nm > >](#)

### Namespaces

- namespace [std](#)

## Macros

- `#define __glibcxx_want_array_constexpr`
- `#define __glibcxx_want_freestanding_array`
- `#define __glibcxx_want_nonmember_container_access`
- `#define __glibcxx_want_to_array`
- `#define _GLIBCXX_ARRAY`

## Functions

- `template<typename _Tp, typename... _Up>  
std::array ( _Tp, _Up...) -> array< enable_if_t<(is_same_v< _Tp, _Up > &&...), _Tp >, 1+sizeof...(_Up)>`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp && std::get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp & std::get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp && std::get (const array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp & std::get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<typename _Tp, size_t _Nm>  
constexpr __detail::__synth3way_t< _Tp > std::operator<=> (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>  
constexpr bool std::operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
__enable_if_t<!__array_traits< _Tp, _Nm >::is_swappable::value > std::swap (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete`
- `template<typename _Tp, std::size_t _Nm>  
constexpr __enable_if_t< __array_traits< _Tp, _Nm >::is_swappable::value > std::swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))`

## Variables

- `template<typename _Tp, size_t _Nm>  
constexpr size_t std::tuple_size_v< array< _Tp, _Nm > >`
- `template<typename _Tp, size_t _Nm>  
constexpr size_t std::tuple_size_v< const array< _Tp, _Nm > >`

### 6.7.1 Detailed Description

This is a Standard C++ Library header.

## 6.8 experimental/array File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define __cpp_lib_experimental_make_array`
- `#define _GLIBCXX_EXPERIMENTAL_ARRAY`

## Functions

- `template<typename _Tp, size_t _Nm, size_t... _Idx>  
constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::\_\_to\_array (_Tp(&__a)[_Nm],  
index\_sequence< _Idx... >)`
- `template<typename _Dest = void, typename... _Types>  
constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::make\_array  
(_Types &&... __t)`
- `template<typename _Tp, size_t _Nm>  
constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::to\_array (_Tp(&__a)[_Nm]) noexcept(is_nothrow_constructible<  
remove_cv_t< _Tp >, _Tp &>::value)`

### 6.8.1 Detailed Description

This is a TS C++ Library header.

## 6.9 atomic File Reference

### Classes

- struct [std::atomic< \\_Tp >](#)
- struct [std::atomic< \\_Tp \\* >](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define \_\_glibcxx\_want\_atomic\_flag\_test`
- `#define \_\_glibcxx\_want\_atomic\_float`
- `#define \_\_glibcxx\_want\_atomic\_is\_always\_lock\_free`
- `#define \_\_glibcxx\_want\_atomic\_lock\_free\_type\_aliases`
- `#define \_\_glibcxx\_want\_atomic\_ref`
- `#define \_\_glibcxx\_want\_atomic\_value\_initialization`
- `#define \_\_glibcxx\_want\_atomic\_wait`
- `#define \_GLIBCXX\_ATOMIC`

### Typedefs

- `typedef atomic< bool > std::atomic\_bool`
- `typedef atomic< char > std::atomic\_char`
- `typedef atomic< char16_t > std::atomic\_char16\_t`
- `typedef atomic< char32_t > std::atomic\_char32\_t`
- `typedef atomic< int > std::atomic\_int`
- `typedef atomic< int16_t > std::atomic\_int16\_t`
- `typedef atomic< int32_t > std::atomic\_int32\_t`
- `typedef atomic< int64_t > std::atomic\_int64\_t`
- `typedef atomic< int8_t > std::atomic\_int8\_t`
- `typedef atomic< int_fast16_t > std::atomic\_int\_fast16\_t`
- `typedef atomic< int_fast32_t > std::atomic\_int\_fast32\_t`
- `typedef atomic< int_fast64_t > std::atomic\_int\_fast64\_t`
- `typedef atomic< int_fast8_t > std::atomic\_int\_fast8\_t`
- `typedef atomic< int_least16_t > std::atomic\_int\_least16\_t`

- typedef `atomic< int_least32_t >` `std::atomic_int_least32_t`
- typedef `atomic< int_least64_t >` `std::atomic_int_least64_t`
- typedef `atomic< int_least8_t >` `std::atomic_int_least8_t`
- typedef `atomic< intmax_t >` `std::atomic_intmax_t`
- typedef `atomic< intptr_t >` `std::atomic_intptr_t`
- typedef `atomic< long long >` `std::atomic_llong`
- typedef `atomic< long >` `std::atomic_long`
- typedef `atomic< ptrdiff_t >` `std::atomic_ptrdiff_t`
- typedef `atomic< signed char >` `std::atomic_schar`
- typedef `atomic< short >` `std::atomic_short`
- typedef `atomic< size_t >` `std::atomic_size_t`
- typedef `atomic< unsigned char >` `std::atomic_uchar`
- typedef `atomic< unsigned int >` `std::atomic_uint`
- typedef `atomic< uint16_t >` `std::atomic_uint16_t`
- typedef `atomic< uint32_t >` `std::atomic_uint32_t`
- typedef `atomic< uint64_t >` `std::atomic_uint64_t`
- typedef `atomic< uint8_t >` `std::atomic_uint8_t`
- typedef `atomic< uint_fast16_t >` `std::atomic_uint_fast16_t`
- typedef `atomic< uint_fast32_t >` `std::atomic_uint_fast32_t`
- typedef `atomic< uint_fast64_t >` `std::atomic_uint_fast64_t`
- typedef `atomic< uint_fast8_t >` `std::atomic_uint_fast8_t`
- typedef `atomic< uint_least16_t >` `std::atomic_uint_least16_t`
- typedef `atomic< uint_least32_t >` `std::atomic_uint_least32_t`
- typedef `atomic< uint_least64_t >` `std::atomic_uint_least64_t`
- typedef `atomic< uint_least8_t >` `std::atomic_uint_least8_t`
- typedef `atomic< uintmax_t >` `std::atomic_uintmax_t`
- typedef `atomic< uintptr_t >` `std::atomic_uintptr_t`
- typedef `atomic< unsigned long long >` `std::atomic_ullong`
- typedef `atomic< unsigned long >` `std::atomic_ulong`
- typedef `atomic< unsigned short >` `std::atomic_ushort`
- typedef `atomic< wchar_t >` `std::atomic_wchar_t`

## Functions

- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_strong** (`atomic< _ITp > *__a`, `__atomic_val_t< _ITp > *__i1`, `__atomic_val_t< _ITp > __i2`) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_strong** (volatile `atomic< _ITp > *__a`, `__atomic_val_t< _ITp > *__i1`, `__atomic_val_t< _ITp > __i2`) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_strong\_explicit** (`atomic< _ITp > *__a`, `__atomic_val_t< _ITp > *__i1`, `__atomic_val_t< _ITp > __i2`, `memory_order __m1`, `memory_order __m2`) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_strong\_explicit** (volatile `atomic< _ITp > *__a`, `__atomic_val_t< _ITp > *__i1`, `__atomic_val_t< _ITp > __i2`, `memory_order __m1`, `memory_order __m2`) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_weak** (`atomic< _ITp > *__a`, `__atomic_val_t< _ITp > *__i1`, `__atomic_val_t< _ITp > __i2`) noexcept
- template<typename \_ITp>  
bool **std::atomic\_compare\_exchange\_weak** (volatile `atomic< _ITp > *__a`, `__atomic_val_t< _ITp > *__i1`, `__atomic_val_t< _ITp > __i2`) noexcept



- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1,`  
`__atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp >`  
`*__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_exchange (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_exchange (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_exchange_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m)`  
`noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_exchange_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i,`  
`memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_add (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_add (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_add_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m)`  
`noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_add_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i,`  
`memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i,`  
`memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i,`  
`memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i,`  
`memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i,`  
`memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m)`  
`noexcept`

- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void std::atomic_flag_clear (atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void std::atomic_init (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void std::atomic_init (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load_explicit (const atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load_explicit (const volatile atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`

### 6.9.1 Detailed Description

This is a Standard C++ Library header.

## 6.10 auto\_ptr.h File Reference

### Classes

- class [std::auto\\_ptr<\\_Tp>](#)
- struct [std::auto\\_ptr\\_ref<\\_Tp1>](#)

### Namespaces

- namespace [std](#)

#### 6.10.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.11 backward\_warning.h File Reference

### 6.11.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 6.12 hash\_fun.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Typedefs

- typedef `__SIZE_TYPE__` [\\_\\_gnu\\_cxx::size\\_t](#)

### Functions

- `size_t` [\\_\\_gnu\\_cxx::\\_\\_stl\\_hash\\_string](#) (const char \* \_\_s)

#### 6.12.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.13 hash\_map File Reference

### Classes

- class [\\_\\_gnu\\_cxx::allocator<\\_Tp>](#)
- struct [\\_\\_gnu\\_cxx::equal\\_to<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::hash\\_map<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#)
- class [\\_\\_gnu\\_cxx::hash\\_multimap<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#)
- struct [\\_\\_gnu\\_cxx::pair<\\_T1, \\_T2>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

**Macros**

- #define **\_BACKWARD\_HASH\_MAP**

**Functions**

- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const [hash\\_map](#)< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm1, const [hash\\_map](#)< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HF, class \_EqKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const [hash\\_multimap](#)< \_Key, \_Tp, \_HF, \_EqKey, \_Alloc > &\_\_hm1, const [hash\\_multimap](#)< \_Key, \_Tp, \_HF, \_EqKey, \_Alloc > &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator==** (const [hash\\_map](#)< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm1, const [hash\\_map](#)< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HF, class \_EqKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator==** (const [hash\\_multimap](#)< \_Key, \_Tp, \_HF, \_EqKey, \_Alloc > &\_\_hm1, const [hash\\_multimap](#)< \_Key, \_Tp, \_HF, \_EqKey, \_Alloc > &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc>  
void **\_\_gnu\_cxx::swap** ([hash\\_map](#)< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm1, [hash\\_map](#)< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc>  
void **\_\_gnu\_cxx::swap** ([hash\\_multimap](#)< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm1, [hash\\_multimap](#)< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm2)

**6.13.1 Detailed Description**

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

**6.14 hash\_set File Reference****Classes**

- class [\\_\\_gnu\\_cxx::hash\\_multiset](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc >
- class [\\_\\_gnu\\_cxx::hash\\_set](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc >

**Namespaces**

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

**Macros**

- #define **\_BACKWARD\_HASH\_SET**

**Functions**

- template<class \_Val, class \_HashFcn, class \_EqualKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const [hash\\_multiset](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, const [hash\\_multiset](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)
- template<class \_Value, class \_HashFcn, class \_EqualKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const [hash\\_set](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, const [hash\\_set](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)

- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool __gnu_cxx::operator== (const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool __gnu_cxx::operator== (const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`void __gnu_cxx::swap (hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`void __gnu_cxx::swap (hash\_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash\_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`

#### 6.14.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

### 6.15 `strstream` File Reference

#### Namespaces

- namespace [std](#)

#### Macros

- `#define _BACKWARD_STRSTREAM`
- `#define _GLIBCXX_STRSTREAM_DEPR(A, B)`

#### 6.15.1 Detailed Description

This is a Standard C++ Library header.

### 6.16 `barrier` File Reference

#### Macros

- `#define __glibcxx_want_barrier`
- `#define _GLIBCXX_BARRIER`

#### 6.16.1 Detailed Description

This is a Standard C++ Library header.

### 6.17 `bit` File Reference

#### Namespaces

- namespace [std](#)

#### Macros

- `#define __glibcxx_want_bit_cast`
- `#define __glibcxx_want_bitops`
- `#define __glibcxx_want_byteswap`
- `#define __glibcxx_want_endian`
- `#define __glibcxx_want_int_pow2`
- `#define _GLIBCXX_BIT`

### 6.17.1 Detailed Description

This is a Standard C++ Library header.

## 6.18 bits/algorithmfwd.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Filter>`  
`constexpr _Filter std::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate>`  
`constexpr _Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter, typename _Tp>`  
`constexpr bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare>`  
`constexpr bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)`
- `template<typename _Iter, typename _OIter>`  
`constexpr _OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1, typename _BIter2>`  
`constexpr _BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`constexpr _OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter>`  
`constexpr _OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp>`  
`constexpr iterator\_traits< _Iter >::difference_type std::count (_Iter, _Iter, const _Tp &)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr iterator\_traits< _Iter >::difference_type std::count_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Filter, typename _Tp>`  
`constexpr pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare>`  
`constexpr pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp>`  
`constexpr void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp>`  
`constexpr _OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp>`  
`constexpr _Iter std::find (_Iter, _Iter, const _Tp &)`

- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr _Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr _Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr _Iter std::find_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr _Iter std::find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct>`  
`constexpr _Funct std::for_each (_Iter, _Iter, _Funct)`
- `template<typename _Filter, typename _Generator>`  
`constexpr void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`constexpr _OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare>`  
`constexpr bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _BIter>`  
`void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare>`  
`void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr _RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr _RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool std::is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr bool std::is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr bool std::is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _Filter>`  
`constexpr bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter>`  
`constexpr _Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr void std::iter_swap (_Filter1, _Filter2)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`

- `template<typename _Iter1, typename _Iter2, typename _Compare>`  
`constexpr bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _Filter, typename _Tp>`  
`constexpr _Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare>`  
`constexpr _Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAIter>`  
`constexpr void std::make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter>`  
`constexpr _Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _Olter>`  
`constexpr _Olter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _Olter)`
- `template<typename _Iter1, typename _Iter2, typename _Olter, typename _Compare>`  
`constexpr _Olter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Compare)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter>`  
`constexpr _Filter std::min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter>`  
`constexpr pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2)`



- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _BIter>`  
`constexpr bool std::next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare>`  
`constexpr bool std::next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool std::none_of (_Iter, _Iter, _Predicate)`
- `template<typename _RAIter>`  
`constexpr void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _RAIter>`  
`constexpr _RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare>`  
`constexpr _RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate>`  
`constexpr _BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate>`  
`constexpr pair< _OIter1, _OIter2 > std::partition_copy (_Iter, _Iter, _OIter1, _OIter2, _Predicate)`
- `template<typename _Filter, typename _Predicate>`  
`constexpr _Filter std::partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _RAIter>`  
`constexpr void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter>`  
`constexpr bool std::prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare>`  
`constexpr bool std::prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr void std::push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter>`  
`void std::random_shuffle (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Generator>`  
`void std::random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _Filter, typename _Tp>`  
`constexpr _Filter std::remove (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp>`  
`constexpr _OIter std::remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`constexpr _OIter std::remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Filter, typename _Predicate>`  
`constexpr _Filter std::remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Tp>`  
`constexpr void std::replace (_Filter, _Filter, const _Tp &, const _Tp &)`

- `template<typename _Iter, typename _OIter, typename _Tp>`  
`constexpr _OIter std::replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp>`  
`constexpr _OIter std::replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`constexpr void std::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _BIter>`  
`constexpr void std::reverse (_BIter, _BIter)`
- `template<typename _BIter, typename _OIter>`  
`constexpr _OIter std::reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _Filter>`  
`constexpr _Filter std::rotate (_Filter, _Filter, _Filter)`
- `template<typename _Filter, typename _OIter>`  
`constexpr _OIter std::rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr _Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter, typename _Size, typename _Tp>`  
`constexpr _Filter std::search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate>`  
`constexpr _Filter std::search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _RAIter, typename _UGenerator>`  
`void std::shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RAIter>`  
`constexpr void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr void std::sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate>`  
`_BIter std::stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter>`  
`void std::stable_sort (_RAIter, _RAIter)`

- `template<typename _RAIter, typename _Compare>`  
`void std::stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter2 std::swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`constexpr _OIter std::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation>`  
`constexpr _OIter std::transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _Filter>`  
`constexpr _Filter std::unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate>`  
`constexpr _Filter std::unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _OIter>`  
`constexpr _OIter std::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate>`  
`constexpr _OIter std::unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _Filter, typename _Tp>`  
`constexpr _Filter std::upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare>`  
`constexpr _Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`

### 6.18.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 6.19 parallel/algorithmfwd.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::parallel](#)

### Functions

- `template<typename _Filter, typename _BiPredicate, typename _IterTag>`  
`_Filter std::parallel::adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _Filter, typename _IterTag>`  
`_Filter std::parallel::adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _RAIter>`  
`_RAIter std::parallel::adjacent_find_switch (_RAIter __begin, _RAIter __end, random\_access\_iterator\_tag)`
- `template<typename _RAIter, typename _BiPredicate>`  
`_RAIter std::parallel::adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag>`  
`iterator\_traits< _Iter >::difference_type std::parallel::count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`iterator\_traits< _RAIter >::difference_type std::parallel::count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag>`  
`iterator\_traits< _Iter >::difference_type std::parallel::count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp>`  
`iterator\_traits< _RAIter >::difference_type std::parallel::count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2>`  
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2>`  
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag>`  
`_RAIter std::parallel::find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random\_access\_iterator\_tag, _IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag>`  
`_Iter std::parallel::find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag>`  
`_Iter std::parallel::find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp>`  
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end, const _Tp & __val, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag>`  
`_Function std::parallel::for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _RAIter, typename _Function>`  
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag>`  
`_OIter std::parallel::generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator>`  
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator, typename _IterTag>`  
`void std::parallel::generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Generator>`  
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`bool std::parallel::lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`bool std::parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag>`  
`_Filter std::parallel::max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag>`  
`_Filter std::parallel::min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch ( _Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _IterTag>`  
`_Filter std::parallel::partition_switch ( _Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter std::parallel::partition_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag>`  
`void std::parallel::replace_if_switch ( _Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp>`  
`void std::parallel::replace_if_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag>`  
`void std::parallel::replace_switch ( _Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp>`  
`void std::parallel::replace_switch ( _RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag>`  
`_Filter std::parallel::search_n_switch ( _Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_RAIter std::parallel::search_n_switch ( _RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2>`  
`_Filter1 std::parallel::search_switch ( _Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2>`  
`_Filter1 std::parallel::search_switch ( _Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2>`  
`_RAIter1 std::parallel::search_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate>`  
`_RAIter1 std::parallel::search_switch ( _RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter std::parallel::set_difference_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter std::parallel::set_difference_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter std::parallel::set_intersection_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter std::parallel::set_intersection_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter std::parallel::set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter std::parallel::set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2>`  
`_OIter std::parallel::transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation>`  
`_RAOIter std::parallel::transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _IterTag3>`  
`_OIter std::parallel::transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation>`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`_OIter std::parallel::unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate>`  
`_RandomAccess_OIter std::parallel::unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter>`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _BiPredicate>`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate>`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp & __value)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp & __value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp & __value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred)`

- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp>`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter>`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Filter>`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate>`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate>`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate>`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Function>`  
`_Function std::__parallel::for_each ( _Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Function>`  
`_Function std::__parallel::for_each ( _Iter, _Iter, _Function)`
- `template<typename _Iterator, typename _Function>`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator>`  
`void std::__parallel::generate ( _Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator>`  
`void std::__parallel::generate ( _Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Generator>`  
`void std::__parallel::generate ( _Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter std::__parallel::generate_n ( _OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter std::__parallel::generate_n ( _OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter std::__parallel::generate_n ( _OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::__parallel::lexicographical_compare ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`

- `template<typename _Iter1, typename _Iter2>`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter std::__parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter>`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter std::__parallel::min_element (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter>`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred)`



- `template<typename _RAIter>`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate>`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Predicate>`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp>`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp>`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Tp>`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2>`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2>`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate>`  
`constexpr _Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate>`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp>`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &)`

- `template<typename _Filter, typename _Integer, typename _Tp>`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter>`  
`_Olter std::__parallel::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter>`  
`_Olter std::__parallel::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate>`  
`_Olter std::__parallel::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate>`  
`_Olter std::__parallel::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter>`  
`_Olter std::__parallel::set_intersection (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter>`  
`_Olter std::__parallel::set_intersection (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate>`  
`_Olter std::__parallel::set_intersection (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate>`  
`_Olter std::__parallel::set_intersection (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter>`  
`_Olter std::__parallel::set_symmetric_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter>`  
`_Olter std::__parallel::set_symmetric_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate>`  
`_Olter std::__parallel::set_symmetric_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate>`  
`_Olter std::__parallel::set_symmetric_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter>`  
`_Olter std::__parallel::set_union (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter>`  
`_Olter std::__parallel::set_union (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate>`  
`_Olter std::__parallel::set_union (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate>`  
`_Olter std::__parallel::set_union (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter>`  
`void std::__parallel::sort (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter>`  
`void std::__parallel::sort (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare>`  
`void std::__parallel::sort (_RAlter __begin, _RAlter __end, _Compare __comp)`
- `template<typename _RAlter, typename _Compare>`  
`void std::__parallel::sort (_RAlter __begin, _RAlter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter>`  
`void std::__parallel::stable_sort (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter>`  
`void std::__parallel::stable_sort (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare>`  
`void std::__parallel::stable_sort (_RAlter __begin, _RAlter __end, _Compare __comp)`

- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter>`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter>`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`

### 6.19.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 6.20 align.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `void * std::align (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept`
- `template<size_t _Align, class _Tp>`  
`constexpr _Tp * std::assume\_aligned (_Tp * __ptr) noexcept`

### 6.20.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.21 bits/alloc\_traits.h File Reference

### Classes

- struct [std::allocator\\_traits](#)< [\\_Alloc](#) >
- struct [std::allocator\\_traits](#)< [allocator](#)< [\\_Tp](#) > >
- struct [std::allocator\\_traits](#)< [allocator](#)< [void](#) > >

### Namespaces

- namespace [std](#)

### 6.21.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.22 ext/alloc\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits<\\_Alloc, typename>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.22.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.23 allocated\_ptr.h File Reference

### Namespaces

- namespace [std](#)

### 6.23.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.24 allocator.h File Reference

### Classes

- class [std::allocator<\\_Tp>](#)

### Namespaces

- namespace [std](#)

### 6.24.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.25 atomic\_base.h File Reference

### Classes

- struct [std::atomic\\_flag](#)

### Namespaces

- namespace [std](#)

## Macros

- #define `_GLIBCXX_ALWAYS_INLINE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_VAR_INIT(_V)`

## Typedefs

- typedef unsigned char `std::__atomic_flag_data_type`

## Enumerations

- enum class `std::memory_order` : int {  
    `relaxed` , `consume` , `acquire` , `release` ,  
    `acq_rel` , `seq_cst` }

## Functions

- void `std::atomic_signal_fence` (`memory_order __m`) noexcept
- void `std::atomic_thread_fence` (`memory_order __m`) noexcept
- template<typename `_Tp`>  
    `_Tp std::kill_dependency` (`_Tp __y`) noexcept
- constexpr `memory_order std::operator&` (`memory_order __m`, `__memory_order_modifier __mod`) noexcept
- constexpr `memory_order std::operator|` (`memory_order __m`, `__memory_order_modifier __mod`) noexcept

## Variables

- constexpr `memory_order std::memory_order_acq_rel`
- constexpr `memory_order std::memory_order_acquire`
- constexpr `memory_order std::memory_order_consume`
- constexpr `memory_order std::memory_order_relaxed`
- constexpr `memory_order std::memory_order_release`
- constexpr `memory_order std::memory_order_seq_cst`

### 6.25.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 6.26 `atomic_futex.h` File Reference

### Namespaces

- namespace `std`

### 6.26.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.27 `atomic_lockfree_defines.h` File Reference

### Macros

- #define `ATOMIC_BOOL_LOCK_FREE`
- #define `ATOMIC_CHAR16_T_LOCK_FREE`

- `#define ATOMIC_CHAR32_T_LOCK_FREE`
- `#define ATOMIC_CHAR_LOCK_FREE`
- `#define ATOMIC_INT_LOCK_FREE`
- `#define ATOMIC_LLONG_LOCK_FREE`
- `#define ATOMIC_LONG_LOCK_FREE`
- `#define ATOMIC_POINTER_LOCK_FREE`
- `#define ATOMIC_SHORT_LOCK_FREE`
- `#define ATOMIC_WCHAR_T_LOCK_FREE`

### 6.27.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 6.28 `atomic_timed_wait.h` File Reference

### 6.28.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 6.29 `atomic_wait.h` File Reference

### 6.29.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 6.30 `basic_ios.h` File Reference

### Classes

- class `std::basic_ios<_CharT, _Traits>`

### Namespaces

- namespace `std`

### Functions

- `template<typename _Facet>`  
`const _Facet & std::__check_facet (const _Facet *__f)`

### 6.30.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.31 `basic_ios.tcc` File Reference

### Namespaces

- namespace `std`

## Macros

- `#define _BASIC_IOS_TCC`

### 6.31.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.32 `basic_string.h` File Reference

### Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
- struct `std::hash<basic_string<char, char_traits<char>, _Alloc>>`
- struct `std::hash<basic_string<char16_t, char_traits<char16_t>, _Alloc>>`
- struct `std::hash<basic_string<char32_t, char_traits<char32_t>, _Alloc>>`
- struct `std::hash<basic_string<wchar_t, char_traits<wchar_t>, _Alloc>>`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Functions

- template<typename \_Str>  
constexpr \_Str **std::str\_concat** (typename \_Str::value\_type const \*\_\_lhs, typename \_Str::size\_type \_\_lhs\_len, typename \_Str::value\_type const \*\_\_rhs, typename \_Str::size\_type \_\_rhs\_len, typename \_Str::allocator\_type const &\_\_a)
- constexpr void **std::\_\_to\_wstring\_numeric** (const char \*\_\_s, int \_\_len, wchar\_t \*\_\_wout)
- **wstring** **std::\_\_to\_wstring\_numeric** (const **string** &\_\_s)
- template<typename \_InputIterator, typename \_CharT = typename iterator\_traits<\_InputIterator>::value\_type, typename \_Allocator = allocator<\_CharT>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>  
**std::basic\_string** (\_InputIterator, \_InputIterator, \_Allocator=\_Allocator()) -> **basic\_string**<\_CharT, **char\_traits**<\_CharT>, \_Allocator>
- template<typename \_CharT, typename \_Traits, typename \_Allocator = allocator<\_CharT>, typename = \_RequireAllocator<\_Allocator>>>  
**std::basic\_string** (**basic\_string\_view**<\_CharT, \_Traits>, const \_Allocator &=\_Allocator()) -> **basic\_string**<\_CharT, \_Traits, \_Allocator>
- template<typename \_CharT, typename \_Traits, typename \_Allocator = allocator<\_CharT>, typename = \_RequireAllocator<\_Allocator>>>  
**std::basic\_string** (**basic\_string\_view**<\_CharT, \_Traits>, typename **basic\_string**<\_CharT, \_Traits, \_Allocator>::size\_type, typename **basic\_string**<\_CharT, \_Traits, \_Allocator>::size\_type, const \_Allocator &=\_Allocator()) -> **basic\_string**<\_CharT, \_Traits, \_Allocator>
- template<typename \_CharT, typename \_Traits, typename \_Alloc>  
**basic\_istream**<\_CharT, \_Traits> & **std::getline** (**basic\_istream**<\_CharT, \_Traits> &&\_\_is, **basic\_string**<\_CharT, \_Traits, \_Alloc> &\_\_str)
- template<typename \_CharT, typename \_Traits, typename \_Alloc>  
**basic\_istream**<\_CharT, \_Traits> & **std::getline** (**basic\_istream**<\_CharT, \_Traits> &&\_\_is, **basic\_string**<\_CharT, \_Traits, \_Alloc> &\_\_str, \_CharT \_\_delim)
- template<typename \_CharT, typename \_Traits, typename \_Alloc>  
**basic\_istream**<\_CharT, \_Traits> & **std::getline** (**basic\_istream**<\_CharT, \_Traits> &&\_\_is, **basic\_string**<\_CharT, \_Traits, \_Alloc> &\_\_str)
- template<typename \_CharT, typename \_Traits, typename \_Alloc>  
**basic\_istream**<\_CharT, \_Traits> & **std::getline** (**basic\_istream**<\_CharT, \_Traits> &&\_\_is, **basic\_string**<\_CharT, \_Traits, \_Alloc> &\_\_str, \_CharT \_\_delim)

- `template<> basic_istream< char > & std::getline (basic_istream< char > &__in, basic_string< char > &__str, char __delim)`
- `template<> basic_istream< wchar_t > & std::getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr auto std::operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr auto std::operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
constexpr bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`



- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string<`  
`_CharT, _Traits, _Alloc > &__str)`
- `template<> basic_istream< char > & std::operator>> (basic_istream< char > &__is, basic_string< char >`  
`&__str)`
- `double std::stod (const string &__str, size_t * __idx=0)`
- `double std::stod (const wstring &__str, size_t * __idx=0)`
- `float std::stof (const string &__str, size_t * __idx=0)`
- `float std::stof (const wstring &__str, size_t * __idx=0)`
- `int std::stoi (const string &__str, size_t * __idx=0, int __base=10)`
- `int std::stoi (const wstring &__str, size_t * __idx=0, int __base=10)`
- `long std::stol (const string &__str, size_t * __idx=0, int __base=10)`
- `long std::stol (const wstring &__str, size_t * __idx=0, int __base=10)`
- `long double std::stold (const string &__str, size_t * __idx=0)`
- `long double std::stold (const wstring &__str, size_t * __idx=0)`
- `long long std::stoll (const string &__str, size_t * __idx=0, int __base=10)`
- `long long std::stoll (const wstring &__str, size_t * __idx=0, int __base=10)`
- `unsigned long std::stoul (const string &__str, size_t * __idx=0, int __base=10)`
- `unsigned long std::stoul (const wstring &__str, size_t * __idx=0, int __base=10)`
- `unsigned long long std::stoull (const string &__str, size_t * __idx=0, int __base=10)`
- `unsigned long long std::stoull (const wstring &__str, size_t * __idx=0, int __base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc`  
`> &__rhs) noexcept( /*conditional */)`
- `string std::to_string (double __val)`
- `string std::to_string (float __val)`
- `string std::to_string (int __val) noexcept`
- `string std::to_string (long __val) noexcept`
- `string std::to_string (long double __val)`
- `string std::to_string (long long __val)`
- `string std::to_string (unsigned __val) noexcept`
- `string std::to_string (unsigned long __val) noexcept`
- `string std::to_string (unsigned long long __val)`
- `wstring std::to_wstring (double __val)`
- `wstring std::to_wstring (float __val)`
- `wstring std::to_wstring (int __val)`
- `wstring std::to_wstring (long __val)`
- `wstring std::to_wstring (long double __val)`
- `wstring std::to_wstring (long long __val)`
- `wstring std::to_wstring (unsigned __val)`
- `wstring std::to_wstring (unsigned long __val)`
- `wstring std::to_wstring (unsigned long long __val)`

### 6.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 6.33 basic\_string.tcc File Reference

### Namespaces

- namespace `std`

## Macros

- `#define _BASIC_STRING_TCC`
- `#define _GLIBCXX_STRING_CONSTEXPR`

## Functions

- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`

### 6.33.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 6.34 backward/binders.h File Reference

### Classes

- class `std::binder1st< _Operation >`
- class `std::binder2nd< _Operation >`

### Namespaces

- namespace `std`

### Functions

- `template<typename _Operation, typename _Tp>`  
`binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp>`  
`binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &__x)`

### 6.34.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.35 boost\_concept\_check.h File Reference

### Namespaces

- namespace `__gnu_cxx`
- namespace `__gnu_debug`
- namespace `std`

### Macros

- `#define _GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_var3, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)`

- `#define _GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define _GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT(_OP, _NAME)`
- `#define _IsUnused`

## Functions

- `template<class _Tp>`  
`void __gnu_cxx::__aux_require_boolean_expr (const _Tp &__t)`
- `void __gnu_cxx::__error_type_must_be_a_signed_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_unsigned_integer_type ()`
- `template<class _Concept>`  
`constexpr void __gnu_cxx::__function_requires ()`

### 6.35.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.36 `c++0x_warning.h` File Reference

### 6.36.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 6.37 `char_traits.h` File Reference

### Classes

- `struct __gnu_cxx::Char_types< _CharT >`
- `struct __gnu_cxx::char_traits< _CharT >`
- `struct std::char_traits< _CharT >`
- `struct std::char_traits< wchar_t >`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`
- namespace `std::__detail`

### Macros

- `#define _GLIBCXX_ALWAYS_INLINE`

### Functions

- `template<typename _ChTraits>`  
`constexpr auto std::__detail::__char_traits_cmp_cat (int __cmp) noexcept`

### 6.37.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 6.38 charconv.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<typename _Tp>`  
constexpr void [std::\\_\\_detail::\\_\\_to\\_chars\\_10\\_impl](#) (char \*\_\_first, unsigned \_\_len, \_Tp \_\_val) noexcept
- `template<typename _Tp>`  
constexpr unsigned [std::\\_\\_detail::\\_\\_to\\_chars\\_len](#) (\_Tp \_\_value, int \_\_base) noexcept

#### 6.38.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<charconv>`.

## 6.39 chrono.h File Reference

### Classes

- struct [std::common\\_type< chrono::duration< \\_Rep, \\_Period > >](#)
- struct [std::common\\_type< chrono::duration< \\_Rep, \\_Period >, chrono::duration< \\_Rep, \\_Period > >](#)
- struct [std::common\\_type< chrono::duration< \\_Rep1, \\_Period1 >, chrono::duration< \\_Rep2, \\_Period2 > >](#)
- struct [std::common\\_type< chrono::time\\_point< \\_Clock, \\_Duration > >](#)
- struct [std::common\\_type< chrono::time\\_point< \\_Clock, \\_Duration >, chrono::time\\_point< \\_Clock, \\_Duration > >](#)
- struct [std::common\\_type< chrono::time\\_point< \\_Clock, \\_Duration1 >, chrono::time\\_point< \\_Clock, \\_Duration2 > >](#)
- class [std::chrono::duration< \\_Rep, \\_Period >](#)
- struct [std::chrono::duration\\_values< \\_Rep >](#)
- struct [std::chrono::steady\\_clock](#)
- struct [std::chrono::system\\_clock](#)
- class [std::chrono::time\\_point< \\_Clock, \\_Dur >](#)
- struct [std::chrono::treat\\_as\\_floating\\_point< \\_Rep >](#)

### Namespaces

- namespace [std](#)
- namespace [std::chrono](#)
- namespace [std::filesystem](#)
- namespace [std::literals](#)
- namespace [std::literals::chrono\\_literals](#)

### Typedefs

- using [std::chrono::days](#)
- using [std::chrono::file\\_clock](#)
- `template<typename _Duration>`  
using [std::chrono::file\\_time](#)
- using [std::chrono::high\\_resolution\\_clock](#)
- using [std::chrono::hours](#)
- using [std::chrono::microseconds](#)
- using [std::chrono::milliseconds](#)

- using `std::chrono::minutes`
- using `std::chrono::months`
- using `std::chrono::nanoseconds`
- using `std::chrono::seconds`
- using `std::chrono::sys_days`
- using `std::chrono::sys_seconds`
- template<typename \_Duration>  
using `std::chrono::sys_time`
- using `std::chrono::weeks`
- using `std::chrono::years`

## Functions

- template<typename \_Rep, typename \_Period>  
constexpr `enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > >` `std::chrono::abs`  
(`duration< _Rep, _Period > __d`)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr `__enable_if_is_duration< _ToDur >` `std::chrono::__detail::ceil` (const `duration< _Rep, _Period > &__d`)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr `__enable_if_is_duration< _ToDur >` `std::chrono::ceil` (const `duration< _Rep, _Period > &__d`)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr `enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > >` `std::chrono::ceil` (const `time_point< _Clock, _Dur > &__tp`)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr `__enable_if_is_duration< _ToDur >` `std::chrono::duration_cast` (const `duration< _Rep, _Period > &__d`)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr `__enable_if_is_duration< _ToDur >` `std::chrono::floor` (const `duration< _Rep, _Period > &__d`)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr `enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > >` `std::chrono::floor` (const `time_point< _Clock, _Dur > &__tp`)
- template<char... \_Digits>  
constexpr `chrono::hours std::literals::chrono_literals::operator""h` ()
- constexpr `chrono::duration< long double, ratio< 3600, 1 > >` `std::literals::chrono_literals::operator""h` (long double `__hours`)
- template<char... \_Digits>  
constexpr `chrono::minutes std::literals::chrono_literals::operator""min` ()
- constexpr `chrono::duration< long double, ratio< 60, 1 > >` `std::literals::chrono_literals::operator""min` (long double `__mins`)
- template<char... \_Digits>  
constexpr `chrono::milliseconds std::literals::chrono_literals::operator""ms` ()
- constexpr `chrono::duration< long double, milli >` `std::literals::chrono_literals::operator""ms` (long double `__msecs`)
- template<char... \_Digits>  
constexpr `chrono::nanoseconds std::literals::chrono_literals::operator""ns` ()
- constexpr `chrono::duration< long double, nano >` `std::literals::chrono_literals::operator""ns` (long double `__nsecs`)
- template<char... \_Digits>  
constexpr `chrono::seconds std::literals::chrono_literals::operator""s` ()
- constexpr `chrono::duration< long double >` `std::literals::chrono_literals::operator""s` (long double `__secs`)
- template<char... \_Digits>  
constexpr `chrono::microseconds std::literals::chrono_literals::operator""us` ()

- constexpr `chrono::duration`< long double, `micro` > `std::literals::chrono_literals::operator""us` (long double \_\_↵ usecs)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr `enable_if_t`< \_\_and\_< \_\_is\_duration< \_ToDur >, \_\_not\_< `treat_as_floating_point`< typename \_To↵ Dur::rep > >::value, \_ToDur > `std::chrono::round` (const `duration`< \_Rep, \_Period > &\_\_d)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr `enable_if_t`< \_\_is\_duration\_v< \_ToDur > &&!`treat_as_floating_point_v`< typename \_ToDur::rep >, `time_point`< \_Clock, \_ToDur > > `std::chrono::round` (const `time_point`< \_Clock, \_Dur > &\_\_tp)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr \_\_enable\_if\_t< \_\_is\_duration< \_ToDur >::value, `time_point`< \_Clock, \_ToDur > > `std::chrono::time_point_cast` (const `time_point`< \_Clock, \_Dur > &\_\_t)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr `common_type`< `duration`< \_Rep1, \_Period1 >, `duration`< \_Rep2, \_Period2 > >::type `std::chrono::operator-` (const `duration`< \_Rep1, \_Period1 > &\_\_lhs, const `duration`< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2>  
constexpr `duration`< \_\_common\_rep\_t< \_Rep1, \_\_disable\_if\_is\_duration< \_Rep2 > >, \_Period > `std::chrono::operator%` (const `duration`< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr `common_type`< `duration`< \_Rep1, \_Period1 >, `duration`< \_Rep2, \_Period2 > >::type `std::chrono::operator%` (const `duration`< \_Rep1, \_Period1 > &\_\_lhs, const `duration`< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Rep2, typename \_Period>  
constexpr `duration`< \_\_common\_rep\_t< \_Rep2, \_Rep1 >, \_Period > `std::chrono::operator*` (const \_Rep1 &\_\_s, const `duration`< \_Rep2, \_Period > &\_\_d)
- template<typename \_Rep1, typename \_Period, typename \_Rep2>  
constexpr `duration`< \_\_common\_rep\_t< \_Rep1, \_\_disable\_if\_is\_duration< \_Rep2 > >, \_Period > `std::chrono::operator/` (const `duration`< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr `common_type`< \_Rep1, \_Rep2 >::type `std::chrono::operator/` (const `duration`< \_Rep1, \_Period1 > &\_\_lhs, const `duration`< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool `std::chrono::operator<` (const `duration`< \_Rep1, \_Period1 > &\_\_lhs, const `duration`< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool `std::chrono::operator<=` (const `duration`< \_Rep1, \_Period1 > &\_\_lhs, const `duration`< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
requires three\_way\_comparable<`common_type_t`< \_Rep1, \_Rep2>>  
constexpr auto `std::chrono::operator<=>` (const `duration`< \_Rep1, \_Period1 > &\_\_lhs, const `duration`< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool `std::chrono::operator>` (const `duration`< \_Rep1, \_Period1 > &\_\_lhs, const `duration`< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool `std::chrono::operator>=` (const `duration`< \_Rep1, \_Period1 > &\_\_lhs, const `duration`< \_Rep2, \_Period2 > &\_\_rhs)

- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>`  
`constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type >`  
`std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__`  
`__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >`  
`std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__`  
`__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 > &__`  
`__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator< (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator<= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _`  
`Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>`  
`constexpr auto std::chrono::operator<=> (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _`  
`Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator> (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator>= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _`  
`Clock, _Dur2 > &__rhs)`

## Variables

- `template<typename _Tp, typename = void>`  
`constexpr bool std::chrono::is_clock_v`
- `template<typename _Tp>`  
`constexpr bool std::chrono::is_clock_v< _Tp, void_t< typename _Tp::rep, typename _Tp::period, type-`  
`name _Tp::duration, typename _Tp::time_point::duration, decltype(_Tp::is_steady), decltype(_Tp::`  
`now())> >`
- `template<> constexpr bool std::chrono::is_clock_v< file_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< steady_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< system_clock >`
- `template<typename _Rep>`  
`constexpr bool std::chrono::treat_as_floating_point_v`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< double >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< float >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< int >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long double >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long long >`

### 6.39.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<chrono>`.

### 6.39.2 Function Documentation

#### ceil()

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::ceil (
    const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type `ToDur` and round up.

If the duration cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

#### Template Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>_ToDur</code> | The result type must be a duration. |
|---------------------|-------------------------------------|

#### Parameters

|                  |             |
|------------------|-------------|
| <code>__d</code> | A duration. |
|------------------|-------------|

#### Returns

The value of `__d` converted to type `_ToDur`.

#### Since

C++17

## 6.40 chrono\_io.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::chrono](#)

### Functions

- void **std::chrono::\_\_detail::from\_stream** ()=delete
- template<typename \_CharT, typename \_Traits, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > & \_\_is, const \_CharT \* \_\_fmt, day & \_\_d, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \* \_\_abbrev=nullptr, [minutes](#) \* \_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Rep, typename \_Period, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > & \_\_is, const \_CharT \* \_\_fmt, [duration](#)< \_Rep, \_Period > & \_\_d, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \* \_\_abbrev=nullptr, [minutes](#) \* \_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > & \_\_is, const \_CharT \* \_\_fmt, [file\\_time](#)< \_Duration > & \_\_tp, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \* \_\_abbrev=nullptr, [minutes](#) \* \_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > & \_\_is, const \_CharT \* \_\_fmt, [gps\\_time](#)< \_Duration > & \_\_tp, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \* \_\_abbrev=nullptr, [minutes](#) \* \_\_offset=nullptr)



- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, local_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr,`  
`minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is,`  
`const _CharT *__fmt, month &__m, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, month_day &__md, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, sys_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr,`  
`minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is,`  
`const _CharT *__fmt, tai_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr,`  
`minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, utc_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr,`  
`minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, weekday &__wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, year &__y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, year_month &__ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT *__fmt, year_month_day &__ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes`  
`*__offset=nullptr)`
- `template<typename _Duration>`  
`__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (local_time< _Duration > __time, const`  
`string *__abbrev=nullptr, const seconds *__offset_sec=nullptr)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const day &__d)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const file_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const gps_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const hh_mm_ss< _Duration > &__hms)`

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info &__li)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`requires requires(const sys_time<_Duration>& __st) { __os << __st; }`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_time< _Duration > &__lt)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month &__m)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_day &__md)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_day_last &__mdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_weekday &__mwd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_weekday_last &__mwdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_days &__dp)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_info &__i)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`requires (!treat_as_floating_point_v<typename _Duration::rep>) && ratio_less_v<typename _Duration::period, days::period>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_time< _Duration > &__tp)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const tai_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const utc_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const weekday &__wd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const weekday_indexed &__wdi)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const weekday_last &__wdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const year &__y)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const year_month &__ym)`

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_day &__ymd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_day_last &__ymdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_weekday &__ymwd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_weekday_last &__ymwdl)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const zoned_time< _Duration, _TimeZonePtr > &__t)`
- `template<typename _CharT, typename _Traits, typename _Rep, typename _Period>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const duration< _Rep, _Period > &__d)`
- `template<typename _CharT, __detail::__parsable< _CharT > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__`  
`parsable< _CharT, _Traits, _StrT > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, basic_string< _CharT, _Traits, _Alloc > & __`  
`__abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, basic_string< _CharT, _Traits, _Alloc > & __`  
`__abbrev, minutes & __offset)`
- `template<typename _CharT, typename _Traits = char_traits< _CharT>, typename _StrT = basic_string< _CharT, _Traits>, __detail::__`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, minutes & __offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>`  
`auto std::chrono::parse (const basic_string< _CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__`  
`parsable< _CharT, _Traits, _StrT > _Parsable>`  
`auto std::chrono::parse (const basic_string< _CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp, basic_string<`  
`_CharT, _Traits, _Alloc > & __abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto std::chrono::parse (const basic_string< _CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp, basic_string<`  
`_CharT, _Traits, _Alloc > & __abbrev, minutes & __offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits>, __detail::__parsable<`  
`_CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto std::chrono::parse (const basic_string< _CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp, minutes & __`  
`__offset)`

### 6.40.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<chrono>`.

## 6.41 `codecvt.h` File Reference

### Classes

- class `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>`
- class `std::codecvt<_InternT, _ExternT, _StateT>`
- class `std::codecvt<char, char, mbstate_t>`
- class `std::codecvt<char16_t, char, mbstate_t>`
- class `std::codecvt<char32_t, char, mbstate_t>`
- class `std::codecvt<wchar_t, char, mbstate_t>`
- class `std::codecvt_base`
- class `std::codecvt_byname<_InternT, _ExternT, _StateT>`

### Namespaces

- namespace `std`

### Functions

- template bool `std::has_facet<codecvt<char, char, mbstate_t>>` (const `locale` &)
- template bool `std::has_facet<codecvt<wchar_t, char, mbstate_t>>` (const `locale` &)
- template const `codecvt<char, char, mbstate_t> & std::use_facet<codecvt<char, char, mbstate_t>>` (const `locale` &)
- template const `codecvt<wchar_t, char, mbstate_t> & std::use_facet<codecvt<wchar_t, char, mbstate_t>>` (const `locale` &)

### Variables

- template<typename `_InternT`, typename `_ExternT`, typename `_StateT`>  
`locale::id std::codecvt<_InternT, _ExternT, _StateT>::id`

#### 6.41.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.42 `concept_check.h` File Reference

### Macros

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

#### 6.42.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.43 `cow_string.h` File Reference

### Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Operation>  
void std::basic_string<_CharT, _Traits, _Alloc >__resize_and_overwrite (const size\_type __n, _Operation __op)`

## Variables

- `template<typename _CharT, typename _Traits, typename _Alloc>  
const basic\_string<_CharT, _Traits, _Alloc >::size_type std::basic_string<_CharT, _Traits, _Alloc >::npos`

### 6.43.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Defines the reference-counted COW string implementation.

## 6.44 `cpp_type_traits.h` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define __INT_N(TYPE)`

### Functions

- `template<typename _Iterator>  
constexpr _Iterator std::__miter_base (_Iterator __it)`

### Variables

- `template<typename _ValT, typename _Tp>  
constexpr bool std::__can_use_memchr_for_find`

### 6.44.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/type_traits.h>`.

## 6.45 `cpyfunc_impl.h` File Reference

### Classes

- class [std::copyable\\_function](#)<\_Res(\_ArgTypes...) [\\_GLIBCXX\\_MOF\\_CV](#) noexcept(\_Noex)>

### Namespaces

- namespace [std](#)

**Macros**

- `#define _GLIBCXX_MOF_CV_REF`
- `#define _GLIBCXX_MOF_INV_QUALS`
- `#define _GLIBCXX_MOF_REF`

**6.45.1 Detailed Description**

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

**6.46 cxxabi\_forced.h File Reference****Classes**

- class `__cxxabiv1::__forced_unwind`

**6.46.1 Detailed Description**

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

**6.47 cxxabi\_init\_exception.h File Reference****Namespaces**

- namespace `std`

**Macros**

- `#define _GLIBCXX_CDTOR_CALLABI`
- `#define _GLIBCXX_HAVE_CDTOR_CALLABI`

**Functions**

- `void * __cxxabiv1::__cxa_allocate_exception (size_t) noexcept`
- `void __cxxabiv1::__cxa_free_exception (void *) noexcept`
- `__cxa_refcounted_exception * __cxxabiv1::__cxa_init_primary_exception (void *__object, std::type\_info * <→ __tinfo, void(* __dest)(void *)) noexcept`

**6.47.1 Detailed Description**

This is an internal header file, included by other library headers. Do not attempt to use it directly.

**6.48 deque.tcc File Reference****Namespaces**

- namespace `std`

**Macros**

- `#define _DEQUE_TCC`

## Functions

- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>  
::Deque_iterator< _OTp, _OTp &, _OTp * > std::copy_move_a1 (::Deque_iterator< _ITp, _IRef, _IPtr > __first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>  
_OI std::copy_move_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Tp>  
__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, ::Deque_iterator< _Tp, _Tp &, _Tp * > >::__type std::copy_move_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _CharT>  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ::Deque_iterator< _CharT, _CharT &, _CharT * > >::__type std::copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT > > __first, istreambuf_iterator< _CharT, char_traits< _CharT > > __last, ::Deque_iterator< _CharT, _CharT &, _CharT * > __result)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>  
::Deque_iterator< _OTp, _OTp &, _OTp * > std::copy_move_backward_a1 (::Deque_iterator< _ITp, _IRef, _IPtr > __first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>  
_OI std::copy_move_backward_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Tp>  
__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, ::Deque_iterator< _Tp, _Tp &, _Tp * > >::__type std::copy_move_backward_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>  
_OI std::copy_move_backward_dit (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>  
_OI std::copy_move_dit (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<typename _CharT, typename _Size>  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ::Deque_iterator< _CharT, _CharT &, _CharT * > >::__type std::copy_n_a (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size, ::Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II>  
__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type std::equal_aux1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>  
bool std::equal_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)`
- `template<typename _II, typename _Tp, typename _Ref, typename _Ptr>  
__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type std::equal_aux1 (_II __first1, _II __last1, ::Deque_iterator< _Tp, _Ref, _Ptr > __first2)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II>  
bool std::equal_dit (const ::Deque_iterator< _Tp, _Ref, _Ptr > &__first1, const ::Deque_iterator< _Tp, _Ref, _Ptr > &__last1, _II __first2)`
- `template<typename _Tp, typename _VTP>  
void std::fill_a1 (const ::Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTP &__value)`

- `template<typename _Tp1, typename _Ref, typename _Ptr, typename _Tp2>`  
`int std::__lex_cmp_dit (::Deque_iterator< _Tp1, _Ref, _Ptr > __first1, ::Deque_iterator< _Tp1, _Ref, _Ptr >`  
`__last1, const _Tp2 * __first2, const _Tp2 * __last2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_↵`  
`iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::Deque_iterator<`  
`_Tp2, _Ref2, _Ptr2 > __last2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>`  
`bool std::__lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_↵`  
`iterator< _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 * __first2, _Tp2 * __last2)`
- `template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__lexicographical_compare_aux1 (_Tp1 * __first1, _Tp1 * __last1, ::Deque_iterator< _Tp2, _Ref2,`  
`_Ptr2 > __first2, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)`

### 6.48.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

## 6.49 enable\_special\_members.h File Reference

### Namespaces

- namespace [std](#)

### 6.49.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.50 erase\_if.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<typename _Container, typename _UnsafeContainer, typename _Predicate>`  
`_Container::size_type std::__detail::__erase_nodes_if (_Container & __cont, _UnsafeContainer & __ucont, _↵`  
`Predicate __pred)`

### 6.50.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.51 exception.h File Reference

### Classes

- class [std::exception](#)

### Namespaces

- namespace [std](#)



### 6.51.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.52 `exception_defines.h` File Reference

### Macros

- `#define __catch(X)`
- `#define __throw_exception_again`
- `#define __try`

### 6.52.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 6.53 `exception_ptr.h` File Reference

### Classes

- class `std::__unspecified__::exception_ptr`
- class `std::exception_ptr`

### Namespaces

- namespace `std`

### Functions

- `exception_ptr std::current_exception () noexcept`
- `template<typename _Ex>  
exception_ptr std::make_exception_ptr (_Ex __ex) noexcept`
- `void std::__unspecified__::rethrow_exception (exception_ptr)`
- `void std::rethrow_exception (exception_ptr)`
- `void std::swap (exception_ptr &__lhs, exception_ptr &__rhs)`

### 6.53.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

### 6.53.2 Function Documentation

#### `rethrow_exception()`

```
void std::rethrow_exception (  
    exception_ptr )
```

Throw the object pointed to by the `exception_ptr`.

## 6.54 `formatwd.h` File Reference

### 6.54.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<format>`.

## 6.55 forward\_list.h File Reference

### Classes

- struct [std::\\_Fwd\\_list\\_base<\\_Tp, \\_Alloc>](#)
- struct [std::\\_Fwd\\_list\\_const\\_iterator<\\_Tp>](#)
- struct [std::\\_Fwd\\_list\\_iterator<\\_Tp>](#)
- struct [std::\\_Fwd\\_list\\_node<\\_Tp>](#)
- struct [std::\\_Fwd\\_list\\_node\\_base](#)
- class [std::\\_\\_fwdlst::Iterator<\\_Const, \\_Ptr>](#)
- struct [std::\\_\\_fwdlst::Node<\\_ValPtr>](#)
- struct [std::\\_\\_fwdlst::Node\\_base<\\_VoidPtr>](#)
- class [std::forward\\_list<\\_Tp, \\_Alloc>](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define GLIBCXX_FWDLIST_REMOVE_RETURN_TYPE_TAG`
- `#define GLIBCXX_USE_ALLOC_PTR_FOR_FWD_LIST`

### Functions

- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::forward_list(_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list<_ValT, _Allocator>`
- `template<typename _Tp, typename _Alloc>`  
`__detail::__synth3way_t<_Tp> std::operator<=> (const forward_list<_Tp, _Alloc> &__x, const forward_list<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::operator== (const forward_list<_Tp, _Alloc> &__lx, const forward_list<_Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc>`  
`void std::swap (forward_list<_Tp, _Alloc> &__lx, forward_list<_Tp, _Alloc> &__ly) noexcept(noexcept(__lx.swap(__ly)))`

#### 6.55.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

## 6.56 forward\_list.tcc File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define FORWARD_LIST_TCC`
- `#define GLIBCXX20_ONLY(__expr)`

## Functions

- `template<typename _Tp, typename _Alloc>`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`

### 6.56.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

## 6.57 bits/fs\_dir.h File Reference

### Classes

- class `std::filesystem::directory_entry`
- class `std::filesystem::directory_iterator`
- class `std::filesystem::file_status`
- class `std::filesystem::recursive_directory_iterator`

### Namespaces

- namespace `std`
- namespace `std::filesystem`

### Variables

- `template<> constexpr bool std::ranges::enable_borrowed_range< filesystem::directory_iterator >`
- `template<> constexpr bool std::ranges::enable_borrowed_range< filesystem::recursive_directory_iterator >`
- `template<> constexpr bool std::ranges::enable_view< filesystem::directory_iterator >`
- `template<> constexpr bool std::ranges::enable_view< filesystem::recursive_directory_iterator >`

### 6.57.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 6.58 experimental/bits/fs\_dir.h File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Functions

- `directory_iterator std::experimental::filesystem::begin (directory_iterator __iter) noexcept`
- `recursive_directory_iterator std::experimental::filesystem::begin (recursive_directory_iterator __iter) noexcept`
- `directory_iterator std::experimental::filesystem::end (directory_iterator) noexcept`
- `recursive_directory_iterator std::experimental::filesystem::end (recursive_directory_iterator) noexcept`
- `bool std::experimental::filesystem::operator!= (const directory_iterator &__lhs, const directory_iterator &__rhs)`
- `bool std::experimental::filesystem::operator!= (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)`

- bool **std::experimental::filesystem::operator==** (const directory\_iterator &\_\_lhs, const directory\_iterator &\_\_rhs)
- bool **std::experimental::filesystem::operator==** (const recursive\_directory\_iterator &\_\_lhs, const recursive\_directory\_iterator &\_\_rhs)

### 6.58.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 6.59 bits/fs\_fwd.h File Reference

### Classes

- struct [std::filesystem::space\\_info](#)

### Namespaces

- namespace [std](#)
- namespace [std::filesystem](#)

### Typedefs

- using [std::filesystem::file\\_time\\_type](#)

### Enumerations

- enum class [std::filesystem::copy\\_options](#) : unsigned short { **none**, **skip\_existing**, **overwrite\_existing**, **update\_existing**, **recursive**, **copy\_symlinks**, **skip\_symlinks**, **directories\_only**, **create\_symlinks**, **create\_hard\_links** }
- enum class [std::filesystem::directory\\_options](#) : unsigned char { **none**, **follow\_directory\_symlink**, **skip\_permission\_denied** }
- enum class [std::filesystem::file\\_type](#) : signed char { **none**, **not\_found**, **regular**, **directory**, **symlink**, **block**, **character**, **fifo**, **socket**, **unknown** }
- enum class [std::filesystem::perm\\_options](#) : unsigned { **replace**, **add**, **remove**, **nofollow** }
- enum class [std::filesystem::perms](#) : unsigned { **none**, **owner\_read**, **owner\_write**, **owner\_exec**, **owner\_all**, **group\_read**, **group\_write**, **group\_exec**, **group\_all**, **others\_read**, **others\_write**, **others\_exec**, **others\_all**, **all**, **set\_uid**, **set\_gid**, **sticky\_bit**, **mask**, **unknown** }

### Functions

- void **std::filesystem::copy** (const [path](#) &\_\_from, const [path](#) &\_\_to, [copy\\_options](#) \_\_options)
- void **std::filesystem::copy** (const [path](#) &\_\_from, const [path](#) &\_\_to, [copy\\_options](#) \_\_options, [error\\_code](#) &)
- bool **std::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, [copy\\_options](#) \_\_option)
- bool **std::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, [copy\\_options](#) \_\_option, [error\\_code](#) &)
- [path](#) **std::filesystem::current\_path** ()
- bool **std::filesystem::exists** ([file\\_status](#)) noexcept
- uintmax\_t **std::filesystem::file\_size** (const [path](#) &)

- `uintmax_t std::filesystem::file_size` (const `path` &, `error_code` &) noexcept
  - `uintmax_t std::filesystem::hard_link_count` (const `path` &)
  - `uintmax_t std::filesystem::hard_link_count` (const `path` &, `error_code` &) noexcept
  - `bool std::filesystem::is_other` (`file_status`) noexcept
  - `bool std::filesystem::is_regular_file` (`file_status`) noexcept
  - `bool std::filesystem::is_symlink` (`file_status`) noexcept
  - `file_time_type std::filesystem::last_write_time` (const `path` &)
  - `file_time_type std::filesystem::last_write_time` (const `path` &, `error_code` &) noexcept
  - `copy_options & std::filesystem::operator&=` (`copy_options` &\_\_x, `copy_options` \_\_y) noexcept
  - `constexpr copy_options std::filesystem::operator^` (`copy_options` \_\_x, `copy_options` \_\_y) noexcept
  - `copy_options & std::filesystem::operator^=` (`copy_options` &\_\_x, `copy_options` \_\_y) noexcept
  - `constexpr copy_options std::filesystem::operator|` (`copy_options` \_\_x, `copy_options` \_\_y) noexcept
  - `copy_options & std::filesystem::operator|=` (`copy_options` &\_\_x, `copy_options` \_\_y) noexcept
  - `constexpr copy_options std::filesystem::operator~` (`copy_options` \_\_x) noexcept
  - `void std::filesystem::permissions` (const `path` &, `perms`, `perm_options`, `error_code` &) noexcept
  - `path std::filesystem::proximate` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
  - `path std::filesystem::relative` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
  - `bool std::filesystem::remove` (const `path` &, `error_code` &) noexcept
  - `uintmax_t std::filesystem::remove_all` (const `path` &)
  - `uintmax_t std::filesystem::remove_all` (const `path` &, `error_code` &)
  - `file_status std::filesystem::status` (const `path` &)
  - `file_status std::filesystem::status` (const `path` &, `error_code` &) noexcept
  - `bool std::filesystem::status_known` (`file_status`) noexcept
  - `file_status std::filesystem::symlink_status` (const `path` &)
  - `file_status std::filesystem::symlink_status` (const `path` &, `error_code` &) noexcept
- 
- `perms & std::filesystem::operator&=` (`perms` &\_\_x, `perms` \_\_y) noexcept
  - `constexpr perms std::filesystem::operator^` (`perms` \_\_x, `perms` \_\_y) noexcept
  - `perms & std::filesystem::operator^=` (`perms` &\_\_x, `perms` \_\_y) noexcept
  - `constexpr perms std::filesystem::operator|` (`perms` \_\_x, `perms` \_\_y) noexcept
  - `perms & std::filesystem::operator|=` (`perms` &\_\_x, `perms` \_\_y) noexcept
  - `constexpr perms std::filesystem::operator~` (`perms` \_\_x) noexcept
- 
- `perm_options & std::filesystem::operator&=` (`perm_options` &\_\_x, `perm_options` \_\_y) noexcept
  - `constexpr perm_options std::filesystem::operator^` (`perm_options` \_\_x, `perm_options` \_\_y) noexcept
  - `perm_options & std::filesystem::operator^=` (`perm_options` &\_\_x, `perm_options` \_\_y) noexcept
  - `constexpr perm_options std::filesystem::operator|` (`perm_options` \_\_x, `perm_options` \_\_y) noexcept
  - `perm_options & std::filesystem::operator|=` (`perm_options` &\_\_x, `perm_options` \_\_y) noexcept
  - `constexpr perm_options std::filesystem::operator~` (`perm_options` \_\_x) noexcept
- 
- `directory_options & std::filesystem::operator&=` (`directory_options` &\_\_x, `directory_options` \_\_y) noexcept
  - `constexpr directory_options std::filesystem::operator^` (`directory_options` \_\_x, `directory_options` \_\_y) noexcept
  - `directory_options & std::filesystem::operator^=` (`directory_options` &\_\_x, `directory_options` \_\_y) noexcept
  - `constexpr directory_options std::filesystem::operator|` (`directory_options` \_\_x, `directory_options` \_\_y) noexcept
  - `directory_options & std::filesystem::operator|=` (`directory_options` &\_\_x, `directory_options` \_\_y) noexcept
  - `constexpr directory_options std::filesystem::operator~` (`directory_options` \_\_x) noexcept

### 6.59.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 6.60 experimental/bits/fs\_fwd.h File Reference

### Classes

- struct [std::experimental::filesystem::v1::space\\_info](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Typedefs

- using [std::experimental::filesystem::file\\_time\\_type](#)

### Enumerations

- enum class [std::experimental::filesystem::copy\\_options](#) : unsigned short { **none**, **skip\_existing**, **overwrite\_existing**, **update\_existing**, **recursive**, **copy\_symlinks**, **skip\_symlinks**, **directories\_only**, **create\_symlinks**, **create\_hard\_links** }
- enum class [std::experimental::filesystem::directory\\_options](#) : unsigned char { **none**, **follow\_directory\_symlink**, **skip\_permission\_denied** }
- enum class [std::experimental::filesystem::file\\_type](#) : signed char { **none**, **not\_found**, **regular**, **directory**, **symlink**, **block**, **character**, **fifo**, **socket**, **unknown** }
- enum class [std::experimental::filesystem::perms](#) : unsigned { **none**, **owner\_read**, **owner\_write**, **owner\_exec**, **owner\_all**, **group\_read**, **group\_write**, **group\_exec**, **group\_all**, **others\_read**, **others\_write**, **others\_exec**, **others\_all**, **all**, **set\_uid**, **set\_gid**, **sticky\_bit**, **mask**, **unknown**, **add\_perms**, **remove\_perms**, **symlink\_nofollow** }

### Functions

- void [std::experimental::filesystem::copy](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_options)
- void [std::experimental::filesystem::copy](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_options, [error\\_code](#) &) noexcept
- bool [std::experimental::filesystem::copy\\_file](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_option)
- bool [std::experimental::filesystem::copy\\_file](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_option, [error\\_code](#) &)
- [path](#) [std::experimental::filesystem::current\\_path](#) ()
- bool [std::experimental::filesystem::is\\_regular\\_file](#) (file\_status) noexcept
- bool [std::experimental::filesystem::is\\_symlink](#) (file\_status) noexcept
- copy\_options & [std::experimental::filesystem::operator&=](#) (copy\_options &\_\_x, copy\_options \_\_y) noexcept
- constexpr copy\_options [std::experimental::filesystem::operator^](#) (copy\_options \_\_x, copy\_options \_\_y) noexcept
- copy\_options & [std::experimental::filesystem::operator^=](#) (copy\_options &\_\_x, copy\_options \_\_y) noexcept

- constexpr copy\_options **std::experimental::filesystem::operator|** (copy\_options \_\_x, copy\_options \_\_y) noexcept
  - copy\_options & **std::experimental::filesystem::operator|=** (copy\_options &\_\_x, copy\_options \_\_y) noexcept
  - constexpr copy\_options **std::experimental::filesystem::operator~** (copy\_options \_\_x) noexcept
  - file\_status **std::experimental::filesystem::status** (const path &)
  - file\_status **std::experimental::filesystem::status** (const path &, error\_code &) noexcept
  - bool **std::experimental::filesystem::status\_known** (file\_status) noexcept
  - file\_status **std::experimental::filesystem::symlink\_status** (const path &)
  - file\_status **std::experimental::filesystem::symlink\_status** (const path &, error\_code &) noexcept
- 
- perms & **std::experimental::filesystem::operator&=** (perms &\_\_x, perms \_\_y) noexcept
  - constexpr perms **std::experimental::filesystem::operator^** (perms \_\_x, perms \_\_y) noexcept
  - perms & **std::experimental::filesystem::operator^=** (perms &\_\_x, perms \_\_y) noexcept
  - constexpr perms **std::experimental::filesystem::operator|** (perms \_\_x, perms \_\_y) noexcept
  - perms & **std::experimental::filesystem::operator|=** (perms &\_\_x, perms \_\_y) noexcept
  - constexpr perms **std::experimental::filesystem::operator~** (perms \_\_x) noexcept
- 
- directory\_options & **std::experimental::filesystem::operator&=** (directory\_options &\_\_x, directory\_options \_\_y) noexcept
  - constexpr directory\_options **std::experimental::filesystem::operator^** (directory\_options \_\_x, directory\_options \_\_y) noexcept
  - directory\_options & **std::experimental::filesystem::operator^=** (directory\_options &\_\_x, directory\_options \_\_y) noexcept
  - constexpr directory\_options **std::experimental::filesystem::operator|** (directory\_options \_\_x, directory\_options \_\_y) noexcept
  - directory\_options & **std::experimental::filesystem::operator|=** (directory\_options &\_\_x, directory\_options \_\_y) noexcept
  - constexpr directory\_options **std::experimental::filesystem::operator~** (directory\_options \_\_x) noexcept

### 6.60.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 6.61 bits/fs\_ops.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::filesystem](#)

### Functions

- [path](#) **std::filesystem::absolute** (const [path](#) &\_\_p)
- [path](#) **std::filesystem::absolute** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- [path](#) **std::filesystem::canonical** (const [path](#) &\_\_p)
- [path](#) **std::filesystem::canonical** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- void **std::filesystem::copy** (const [path](#) &\_\_from, const [path](#) &\_\_to)
- void **std::filesystem::copy** (const [path](#) &\_\_from, const [path](#) &\_\_to, [copy\\_options](#) \_\_options)
- void **std::filesystem::copy** (const [path](#) &\_\_from, const [path](#) &\_\_to, [copy\\_options](#) \_\_options, [error\\_code](#) &)
- void **std::filesystem::copy** (const [path](#) &\_\_from, const [path](#) &\_\_to, [error\\_code](#) &\_\_ec)

- bool **std::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to)
- bool **std::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, [copy\\_options](#) \_\_option)
- bool **std::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, [copy\\_options](#) \_\_option, [error\\_code](#) &)
- bool **std::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, [error\\_code](#) &\_\_ec)
- void **std::filesystem::copy\_symlink** (const [path](#) &\_\_existing\_symlink, const [path](#) &\_\_new\_symlink)
- void **std::filesystem::copy\_symlink** (const [path](#) &\_\_existing\_symlink, const [path](#) &\_\_new\_symlink, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::create\_directories** (const [path](#) &\_\_p)
- bool **std::filesystem::create\_directories** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- bool **std::filesystem::create\_directory** (const [path](#) &\_\_p)
- bool **std::filesystem::create\_directory** (const [path](#) &\_\_p, const [path](#) &\_\_attributes)
- bool **std::filesystem::create\_directory** (const [path](#) &\_\_p, const [path](#) &\_\_attributes, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::create\_directory** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- void **std::filesystem::create\_directory\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void **std::filesystem::create\_directory\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink, [error\\_code](#) &\_\_ec) noexcept
- void **std::filesystem::create\_hard\_link** (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link)
- void **std::filesystem::create\_hard\_link** (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link, [error\\_code](#) &\_\_ec) noexcept
- void **std::filesystem::create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void **std::filesystem::create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **std::filesystem::current\_path** ()
- void **std::filesystem::current\_path** (const [path](#) &\_\_p)
- void **std::filesystem::current\_path** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **std::filesystem::current\_path** ([error\\_code](#) &\_\_ec)
- bool **std::filesystem::equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2)
- bool **std::filesystem::equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::exists** (const [path](#) &\_\_p)
- bool **std::filesystem::exists** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::exists** ([file\\_status](#)) noexcept
- [uintmax\\_t](#) **std::filesystem::file\_size** (const [path](#) &)
- [uintmax\\_t](#) **std::filesystem::file\_size** (const [path](#) &, [error\\_code](#) &) noexcept
- [uintmax\\_t](#) **std::filesystem::hard\_link\_count** (const [path](#) &)
- [uintmax\\_t](#) **std::filesystem::hard\_link\_count** (const [path](#) &, [error\\_code](#) &) noexcept
- bool **std::filesystem::is\_block\_file** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_block\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_block\_file** ([file\\_status](#) \_\_s) noexcept
- bool **std::filesystem::is\_character\_file** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_character\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_character\_file** ([file\\_status](#) \_\_s) noexcept
- bool **std::filesystem::is\_directory** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_directory** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_directory** ([file\\_status](#) \_\_s) noexcept
- bool **std::filesystem::is\_empty** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_empty** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- bool **std::filesystem::is\_fifo** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_fifo** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_fifo** ([file\\_status](#) \_\_s) noexcept
- bool **std::filesystem::is\_other** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_other** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept



- `bool std::filesystem::is_other (file_status) noexcept`
- `bool std::filesystem::is_regular_file (const path &__p)`
- `bool std::filesystem::is_regular_file (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_regular_file (file_status) noexcept`
- `bool std::filesystem::is_socket (const path &__p)`
- `bool std::filesystem::is_socket (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_socket (file_status __s) noexcept`
- `bool std::filesystem::is_symlink (const path &__p)`
- `bool std::filesystem::is_symlink (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_symlink (file_status) noexcept`
- `file_time_type std::filesystem::last_write_time (const path &)`
- `file_time_type std::filesystem::last_write_time (const path &, error_code &) noexcept`
- `void std::filesystem::last_write_time (const path &__p, file_time_type __new_time)`
- `void std::filesystem::last_write_time (const path &__p, file_time_type __new_time, error_code &__ec) noexcept`
- `void std::filesystem::permissions (const path &, perms, perm_options, error_code &) noexcept`
- `void std::filesystem::permissions (const path &__p, perms __prms, error_code &__ec) noexcept`
- `void std::filesystem::permissions (const path &__p, perms __prms, perm_options __opts=perm_options::replace)`
- `path std::filesystem::proximate (const path &__p, const path &__base, error_code &__ec)`
- `path std::filesystem::proximate (const path &__p, const path &__base=current_path())`
- `path std::filesystem::proximate (const path &__p, error_code &__ec)`
- `path std::filesystem::read_symlink (const path &__p)`
- `path std::filesystem::read_symlink (const path &__p, error_code &__ec)`
- `path std::filesystem::relative (const path &__p, const path &__base, error_code &__ec)`
- `path std::filesystem::relative (const path &__p, const path &__base=current_path())`
- `path std::filesystem::relative (const path &__p, error_code &__ec)`
- `bool std::filesystem::remove (const path &, error_code &) noexcept`
- `bool std::filesystem::remove (const path &__p)`
- `uintmax_t std::filesystem::remove_all (const path &)`
- `uintmax_t std::filesystem::remove_all (const path &, error_code &)`
- `void std::filesystem::rename (const path &__from, const path &__to)`
- `void std::filesystem::rename (const path &__from, const path &__to, error_code &__ec) noexcept`
- `void std::filesystem::resize_file (const path &__p, uintmax_t __size)`
- `void std::filesystem::resize_file (const path &__p, uintmax_t __size, error_code &__ec) noexcept`
- `space_info std::filesystem::space (const path &__p)`
- `space_info std::filesystem::space (const path &__p, error_code &__ec) noexcept`
- `file_status std::filesystem::status (const path &)`
- `file_status std::filesystem::status (const path &, error_code &) noexcept`
- `bool std::filesystem::status_known (file_status) noexcept`
- `file_status std::filesystem::symlink_status (const path &)`
- `file_status std::filesystem::symlink_status (const path &, error_code &) noexcept`
- `path std::filesystem::temp_directory_path ()`
- `path std::filesystem::temp_directory_path (error_code &__ec)`
- `path std::filesystem::weakly_canonical (const path &__p)`
- `path std::filesystem::weakly_canonical (const path &__p, error_code &__ec)`

### 6.61.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 6.62 experimental/bits/fs\_ops.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Functions

- [path std::experimental::filesystem::absolute](#) (const [path](#) &\_\_p, const [path](#) &\_\_base=current\_path())
- [path std::experimental::filesystem::canonical](#) (const [path](#) &\_\_p, const [path](#) &\_\_base, [error\\_code](#) &\_\_ec)
- [path std::experimental::filesystem::canonical](#) (const [path](#) &\_\_p, const [path](#) &\_\_base=current\_path())
- [path std::experimental::filesystem::canonical](#) (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- void [std::experimental::filesystem::copy](#) (const [path](#) &\_\_from, const [path](#) &\_\_to)
- void [std::experimental::filesystem::copy](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_options)
- void [std::experimental::filesystem::copy](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_options, [error\\_code](#) &\_\_ec) noexcept
- void [std::experimental::filesystem::copy](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, [error\\_code](#) &\_\_ec) noexcept
- bool [std::experimental::filesystem::copy\\_file](#) (const [path](#) &\_\_from, const [path](#) &\_\_to)
- bool [std::experimental::filesystem::copy\\_file](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_option)
- bool [std::experimental::filesystem::copy\\_file](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_option, [error\\_code](#) &\_\_ec)
- bool [std::experimental::filesystem::copy\\_file](#) (const [path](#) &\_\_from, const [path](#) &\_\_to, [error\\_code](#) &\_\_ec)
- void [std::experimental::filesystem::copy\\_symlink](#) (const [path](#) &\_\_existing\_symlink, const [path](#) &\_\_new\_↔ symlink)
- void [std::experimental::filesystem::copy\\_symlink](#) (const [path](#) &\_\_existing\_symlink, const [path](#) &\_\_new\_↔ symlink, [error\\_code](#) &\_\_ec) noexcept
- bool [std::experimental::filesystem::create\\_directories](#) (const [path](#) &\_\_p)
- bool [std::experimental::filesystem::create\\_directories](#) (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- bool [std::experimental::filesystem::create\\_directory](#) (const [path](#) &\_\_p)
- bool [std::experimental::filesystem::create\\_directory](#) (const [path](#) &\_\_p, const [path](#) &\_\_attributes)
- bool [std::experimental::filesystem::create\\_directory](#) (const [path](#) &\_\_p, const [path](#) &\_\_attributes, [error\\_code](#) &\_\_ec) noexcept
- bool [std::experimental::filesystem::create\\_directory](#) (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- void [std::experimental::filesystem::create\\_directory\\_symlink](#) (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void [std::experimental::filesystem::create\\_directory\\_symlink](#) (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink, [error\\_code](#) &\_\_ec) noexcept
- void [std::experimental::filesystem::create\\_hard\\_link](#) (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link)
- void [std::experimental::filesystem::create\\_hard\\_link](#) (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link, [error\\_code](#) &\_\_ec) noexcept
- void [std::experimental::filesystem::create\\_symlink](#) (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void [std::experimental::filesystem::create\\_symlink](#) (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink, [error\\_code](#) &\_\_ec) noexcept
- [path std::experimental::filesystem::current\\_path](#) ()
- void [std::experimental::filesystem::current\\_path](#) (const [path](#) &\_\_p)
- void [std::experimental::filesystem::current\\_path](#) (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- [path std::experimental::filesystem::current\\_path](#) ([error\\_code](#) &\_\_ec)
- bool [std::experimental::filesystem::equivalent](#) (const [path](#) &\_\_p1, const [path](#) &\_\_p2)
- bool [std::experimental::filesystem::equivalent](#) (const [path](#) &\_\_p1, const [path](#) &\_\_p2, [error\\_code](#) &\_\_ec) noexcept
- bool [std::experimental::filesystem::exists](#) (const [path](#) &\_\_p)
- bool [std::experimental::filesystem::exists](#) (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool [std::experimental::filesystem::exists](#) (file\_status \_\_s) noexcept

- `uintmax_t std::experimental::filesystem::file_size` (const `path` & \_\_p)
- `uintmax_t std::experimental::filesystem::file_size` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `uintmax_t std::experimental::filesystem::hard_link_count` (const `path` & \_\_p)
- `uintmax_t std::experimental::filesystem::hard_link_count` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_block_file` (const `path` & \_\_p)
- `bool std::experimental::filesystem::is_block_file` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_block_file` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::is_character_file` (const `path` & \_\_p)
- `bool std::experimental::filesystem::is_character_file` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_character_file` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::is_directory` (const `path` & \_\_p)
- `bool std::experimental::filesystem::is_directory` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_directory` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::is_empty` (const `path` & \_\_p)
- `bool std::experimental::filesystem::is_empty` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_fifo` (const `path` & \_\_p)
- `bool std::experimental::filesystem::is_fifo` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_fifo` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::is_other` (const `path` & \_\_p)
- `bool std::experimental::filesystem::is_other` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_other` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::is_regular_file` (const `path` & \_\_p)
- `bool std::experimental::filesystem::is_regular_file` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_regular_file` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::is_socket` (const `path` & \_\_p)
- `bool std::experimental::filesystem::is_socket` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_socket` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::is_symlink` (const `path` & \_\_p)
- `bool std::experimental::filesystem::is_symlink` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `bool std::experimental::filesystem::is_symlink` (file\_status \_\_s) noexcept
- `file_time_type std::experimental::filesystem::last_write_time` (const `path` & \_\_p)
- `file_time_type std::experimental::filesystem::last_write_time` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `void std::experimental::filesystem::last_write_time` (const `path` & \_\_p, file\_time\_type \_\_new\_time)
- `void std::experimental::filesystem::last_write_time` (const `path` & \_\_p, file\_time\_type \_\_new\_time, `error_code` & \_\_ec) noexcept
- `void std::experimental::filesystem::permissions` (const `path` & \_\_p, perms \_\_prms)
- `void std::experimental::filesystem::permissions` (const `path` & \_\_p, perms \_\_prms, `error_code` & \_\_ec) noexcept
- `path std::experimental::filesystem::read_symlink` (const `path` & \_\_p)
- `path std::experimental::filesystem::read_symlink` (const `path` & \_\_p, `error_code` & \_\_ec)
- `bool std::experimental::filesystem::remove` (const `path` & \_\_p)
- `bool std::experimental::filesystem::remove` (const `path` & \_\_p, `error_code` & \_\_ec) noexcept
- `uintmax_t std::experimental::filesystem::remove_all` (const `path` & \_\_p)
- `uintmax_t std::experimental::filesystem::remove_all` (const `path` & \_\_p, `error_code` & \_\_ec)
- `void std::experimental::filesystem::rename` (const `path` & \_\_from, const `path` & \_\_to)
- `void std::experimental::filesystem::rename` (const `path` & \_\_from, const `path` & \_\_to, `error_code` & \_\_ec) noexcept
- `void std::experimental::filesystem::resize_file` (const `path` & \_\_p, uintmax\_t \_\_size)
- `void std::experimental::filesystem::resize_file` (const `path` & \_\_p, uintmax\_t \_\_size, `error_code` & \_\_ec) noexcept
- `space_info std::experimental::filesystem::space` (const `path` & \_\_p)

- `space_info` `std::experimental::filesystem::space` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `file_status` `std::experimental::filesystem::status` (const `path` &\_\_p)
- `file_status` `std::experimental::filesystem::status` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `bool` `std::experimental::filesystem::status_known` (file\_status \_\_s) noexcept
- `file_status` `std::experimental::filesystem::symlink_status` (const `path` &\_\_p)
- `file_status` `std::experimental::filesystem::symlink_status` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `path` `std::experimental::filesystem::system_complete` (const `path` &\_\_p)
- `path` `std::experimental::filesystem::system_complete` (const `path` &\_\_p, `error_code` &\_\_ec)
- `path` `std::experimental::filesystem::temp_directory_path` ()
- `path` `std::experimental::filesystem::temp_directory_path` (`error_code` &\_\_ec)

### 6.62.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 6.63 bits/fs\_path.h File Reference

### Classes

- class `std::filesystem::filesystem_error`
- class `std::filesystem::path::iterator`
- class `std::filesystem::path`

### Namespaces

- namespace `std`
- namespace `std::filesystem`

### Functions

- `size_t` `std::filesystem::hash_value` (const `path` &\_\_p) noexcept

### 6.63.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 6.64 experimental/bits/fs\_path.h File Reference

### Classes

- class `std::experimental::filesystem::v1::basic_string_view<_CharT, _Traits>`
- class `std::experimental::filesystem::v1::filesystem_error`
- class `std::experimental::filesystem::v1::path::iterator`
- class `std::experimental::filesystem::v1::path`

### Namespaces

- namespace `std`
- namespace `std::experimental`

## Functions

- bool **std::experimental::filesystem::operator<** (const [path](#) &\_\_lhs, const [path](#) &\_\_rhs) noexcept
- bool **std::experimental::filesystem::operator==** (const [path](#) &\_\_lhs, const [path](#) &\_\_rhs) noexcept

### 6.64.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 6.65 fstream.tcc File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _FSTREAM_TCC`

### 6.65.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<fstream>`.

## 6.66 funcref\_impl.h File Reference

### Classes

- class [std::function\\_ref< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>](#)

### Namespaces

- namespace [std](#)

### 6.66.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.67 funtexcept.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- void **std::\_\_throw\_bad\_alloc** (void)
- void **std::\_\_throw\_bad\_array\_new\_length** (void)
- void **std::\_\_throw\_bad\_cast** (void)
- void **std::\_\_throw\_bad\_exception** (void)
- void **std::\_\_throw\_bad\_function\_call** ()
- void **std::\_\_throw\_bad\_typeid** (void)
- void **std::\_\_throw\_domain\_error** (const char \*)
- void **std::\_\_throw\_future\_error** (int)

- void **std::\_\_throw\_invalid\_argument** (const char \*)
- void **std::\_\_throw\_ios\_failure** (const char \*)
- void **std::\_\_throw\_ios\_failure** (const char \*, int)
- void **std::\_\_throw\_length\_error** (const char \*)
- void **std::\_\_throw\_logic\_error** (const char \*)
- void **std::\_\_throw\_out\_of\_range** (const char \*)
- void **std::\_\_throw\_out\_of\_range\_fmt** (const char \*,...)
- void **std::\_\_throw\_overflow\_error** (const char \*)
- void **std::\_\_throw\_range\_error** (const char \*)
- void **std::\_\_throw\_runtime\_error** (const char \*)
- void **std::\_\_throw\_system\_error** (int)
- void **std::\_\_throw\_underflow\_error** (const char \*)

### 6.67.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

This header provides support for `-fno-exceptions`.

## 6.68 functional\_hash.h File Reference

### Classes

- struct [std::\\_\\_is\\_fast\\_hash<\\_Hash>](#)
- struct [std::hash<\\_Tp>](#)
- struct [std::hash<\\_Tp\\*>](#)
- struct [std::hash<bool>](#)
- struct [std::hash<char>](#)
- struct [std::hash<char16\\_t>](#)
- struct [std::hash<char32\\_t>](#)
- struct [std::hash<double>](#)
- struct [std::hash<float>](#)
- struct [std::hash<int>](#)
- struct [std::hash<long>](#)
- struct [std::hash<long double>](#)
- struct [std::hash<long long>](#)
- struct [std::hash<short>](#)
- struct [std::hash<signed char>](#)
- struct [std::hash<unsigned char>](#)
- struct [std::hash<unsigned int>](#)
- struct [std::hash<unsigned long>](#)
- struct [std::hash<unsigned long long>](#)
- struct [std::hash<unsigned short>](#)
- struct [std::hash<wchar\\_t>](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define \_Cxx\_hashtable\_define\_trivial\_hash(_Tp)`

## Variables

- `template<typename _Tp, typename = void>  
constexpr bool std::__is_hash_enabled_for`
- `template<typename _Tp>  
constexpr bool std::__is_hash_enabled_for< _Tp, __void_t< decltype(hash< _Tp >()(declval< _Tp >()))>  
>`

### 6.68.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.69 funcwrap.h File Reference

### 6.69.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.70 gslice.h File Reference

### Classes

- class [std::gslice](#)

### Namespaces

- namespace [std](#)

### 6.70.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.71 gslice\_array.h File Reference

### Classes

- class [std::gslice\\_array< \\_Tp >](#)

### Namespaces

- namespace [std](#)

### 6.71.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.72 hash\_bytes.h File Reference

### Namespaces

- namespace [std](#)

## Functions

- `size_t std::Fnv_hash_bytes` (const void \*\_\_ptr, size\_t \_\_len, size\_t \_\_seed)
- `size_t std::Hash_bytes` (const void \*\_\_ptr, size\_t \_\_len, size\_t \_\_seed)

### 6.72.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.73 backward/hashtable.h File Reference

### Namespaces

- namespace `__gnu_cxx`

### Enumerations

- enum { `_S_num_primes` }

## Functions

- `unsigned long __gnu_cxx::__stl_next_prime` (unsigned long \_\_n)
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>`  
`bool __gnu_cxx::operator!=` (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>`  
`bool __gnu_cxx::operator==` (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All>`  
`void __gnu_cxx::swap` (hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht1, hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht2)

### Variables

- `template<typename _PrimeType>`  
`const _PrimeType __gnu_cxx::Hashtable_prime_list< _PrimeType >::__stl_prime_list [_S_num_primes]`

### 6.73.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

## 6.74 bits/hashtable.h File Reference

### Namespaces

- namespace `std`

### 6.74.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.



## 6.75 hashtable\_policy.h File Reference

### Namespaces

- namespace [std](#)

#### 6.75.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

## 6.76 indirect.h File Reference

### 6.76.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.77 indirect\_array.h File Reference

### Classes

- class [std::indirect\\_array<\\_Tp>](#)

### Namespaces

- namespace [std](#)

#### 6.77.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.78 invoke.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>  
constexpr _Up && std::__invfwd (typename remove\_reference<_Tp>::type &__t) noexcept`
- `template<typename _Callable, typename... _Args>  
constexpr __invoke_result<_Callable, _Args...>::type std::__invoke (_Callable &&__fn, _Args &&... __args)  
noexcept(__is_nothrow_invocable<_Callable, _Args...>::value)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>  
constexpr _Res std::__invoke_impl (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)↵  
args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>  
constexpr _Res std::__invoke_impl (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemPtr, typename _Tp>  
constexpr _Res std::__invoke_impl (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _MemPtr, typename _Tp>  
constexpr _Res std::__invoke_impl (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)`

- `template<typename _Res, typename _Fn, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_other, _Fn &&__f, _Args &&... __args)`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`constexpr enable\_if\_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > std::__invoke_r (_Callable &&__fn, _Args &&... __args) noexcept(is_nothrow_invocable_r_v< _Res, _Callable, _Args... >)`

### 6.78.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.79 ios\_base.h File Reference

### Classes

- class [std::ios\\_base](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _NOREPLACE_UNUSED`

### Enumerations

- enum `_ios_Fmtflags` {  
`_S_boolalpha`, `_S_dec`, `_S_fixed`, `_S_hex`,  
`_S_internal`, `_S_left`, `_S_oct`, `_S_right`,  
`_S_scientific`, `_S_showbase`, `_S_showpoint`, `_S_showpos`,  
`_S_skipws`, `_S_unitbuf`, `_S_uppercase`, `_S_adjustfield`,  
`_S_basefield`, `_S_floatfield`, `_S_ios_fmtflags_end`, `_S_ios_fmtflags_max`,  
`_S_ios_fmtflags_min` }
- enum `_ios_iostate` {  
`_S_goodbit`, `_S_badbit`, `_S_eofbit`, `_S_failbit`,  
`_S_ios_iostate_end`, `_S_ios_iostate_max`, `_S_ios_iostate_min` }
- enum `_ios_Openmode` {  
`_S_app`, `_S_ate`, `_S_bin`, `_S_in`,  
`_S_out`, `_S_trunc`, `_S_noreplace`, `_S_ios_openmode_end`,  
`_S_ios_openmode_max`, `_S_ios_openmode_min` }
- enum `_ios_Seekdir` { `_S_beg`, `_S_cur`, `_S_end`, `_S_ios_seekdir_end` }
- enum class [std::io\\_errc](#) { `stream` }

### Functions

- `ios\_base & std::boolalpha (ios\_base &__base)`
- `ios\_base & std::dec (ios\_base &__base)`
- `ios\_base & std::defaultfloat (ios\_base &__base)`
- `ios\_base & std::fixed (ios\_base &__base)`
- `ios\_base & std::hex (ios\_base &__base)`
- `ios\_base & std::hexfloat (ios\_base &__base)`
- `ios\_base & std::internal (ios\_base &__base)`
- `const error\_category & std::iostream\_category () noexcept`

- `ios_base & std::left (ios_base & __base)`
- `error_code std::make_error_code (io_errc __e) noexcept`
- `error_condition std::make_error_condition (io_errc __e) noexcept`
- `ios_base & std::noboolalpha (ios_base & __base)`
- `ios_base & std::noshowbase (ios_base & __base)`
- `ios_base & std::noshowpoint (ios_base & __base)`
- `ios_base & std::noshowpos (ios_base & __base)`
- `ios_base & std::noskipws (ios_base & __base)`
- `ios_base & std::nounitbuf (ios_base & __base)`
- `ios_base & std::nouppercase (ios_base & __base)`
- `ios_base & std::oct (ios_base & __base)`
- `constexpr _ios_Fmtflags std::operator& (_ios_Fmtflags __a, _ios_Fmtflags __b) noexcept`
- `constexpr _ios_lstate std::operator& (_ios_lstate __a, _ios_lstate __b) noexcept`
- `constexpr _ios_Openmode std::operator& (_ios_Openmode __a, _ios_Openmode __b) noexcept`
- `constexpr const _ios_Fmtflags & std::operator&= (_ios_Fmtflags & __a, _ios_Fmtflags __b) noexcept`
- `constexpr const _ios_lstate & std::operator&= (_ios_lstate & __a, _ios_lstate __b) noexcept`
- `constexpr const _ios_Openmode & std::operator&= (_ios_Openmode & __a, _ios_Openmode __b) noexcept`
- `constexpr _ios_Fmtflags std::operator^ (_ios_Fmtflags __a, _ios_Fmtflags __b) noexcept`
- `constexpr _ios_lstate std::operator^ (_ios_lstate __a, _ios_lstate __b) noexcept`
- `constexpr _ios_Openmode std::operator^ (_ios_Openmode __a, _ios_Openmode __b) noexcept`
- `constexpr const _ios_Fmtflags & std::operator^= (_ios_Fmtflags & __a, _ios_Fmtflags __b) noexcept`
- `constexpr const _ios_lstate & std::operator^= (_ios_lstate & __a, _ios_lstate __b) noexcept`
- `constexpr const _ios_Openmode & std::operator^= (_ios_Openmode & __a, _ios_Openmode __b) noexcept`
- `constexpr _ios_Fmtflags std::operator| (_ios_Fmtflags __a, _ios_Fmtflags __b) noexcept`
- `constexpr _ios_lstate std::operator| (_ios_lstate __a, _ios_lstate __b) noexcept`
- `constexpr _ios_Openmode std::operator| (_ios_Openmode __a, _ios_Openmode __b) noexcept`
- `constexpr const _ios_Fmtflags & std::operator|= (_ios_Fmtflags & __a, _ios_Fmtflags __b) noexcept`
- `constexpr const _ios_lstate & std::operator|= (_ios_lstate & __a, _ios_lstate __b) noexcept`
- `constexpr const _ios_Openmode & std::operator|= (_ios_Openmode & __a, _ios_Openmode __b) noexcept`
- `constexpr _ios_Fmtflags std::operator~ (_ios_Fmtflags __a) noexcept`
- `constexpr _ios_lstate std::operator~ (_ios_lstate __a) noexcept`
- `constexpr _ios_Openmode std::operator~ (_ios_Openmode __a) noexcept`
- `ios_base & std::right (ios_base & __base)`
- `ios_base & std::scientific (ios_base & __base)`
- `ios_base & std::showbase (ios_base & __base)`
- `ios_base & std::showpoint (ios_base & __base)`
- `ios_base & std::showpos (ios_base & __base)`
- `ios_base & std::skipws (ios_base & __base)`
- `ios_base & std::unitbuf (ios_base & __base)`
- `ios_base & std::uppercase (ios_base & __base)`

### 6.79.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.80 istream.tcc File Reference

### Namespaces

- namespace `std`

## Macros

- `#define _ISTREAM_TCC`

## Functions

- `template<typename _CharT, typename _Traits>`  
`void std::__istream_extract (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`

### 6.80.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<istream>`.

## 6.81 iterator\_concepts.h File Reference

### Classes

- struct `std::default_sentinel_t`

### Namespaces

- namespace `std`

### Variables

- constexpr `default_sentinel_t std::default_sentinel`

### 6.81.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.82 list.tcc File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX20_ONLY(__expr)`
- `#define _LIST_TCC`

### 6.82.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

## 6.83 locale\_classes.h File Reference

### Classes

- class [std::\\_\\_cxx11::collate<\\_CharT>](#)
- class [std::\\_\\_cxx11::collate\\_byname<\\_CharT>](#)
- class [std::locale::facet](#)
- class [std::locale::id](#)
- class [std::locale](#)

### Namespaces

- namespace [std](#)

### Variables

- [template<typename \\_CharT> locale::id std::collate<\\_CharT>::id](#)

#### 6.83.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.84 locale\_classes.tcc File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_STD_FACET(...)`
- `#define _LOCALE_CLASSES_TCC`

### Functions

- [template<typename \\_Facet> const \\_Facet \\* std::\\_\\_try\\_use\\_facet](#) (const [locale](#) &\_\_loc)
- [template<typename \\_Facet> bool std::has\\_facet](#) (const [locale](#) &\_\_loc)
- [template<typename \\_Facet> const \\_Facet & std::use\\_facet](#) (const [locale](#) &\_\_loc)

#### 6.84.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.85 locale\_conv.h File Reference

### Classes

- class [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>](#)
- class [std::wstring\\_convert<\\_Codecvt, \\_Elem, \\_Wide\\_alloc, \\_Byte\\_alloc>](#)

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

## Functions

- `template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn>`  
`bool std::\_\_do\_str\_codecvt (const _InChar *__first, const _InChar *__last, _OutStr &__outstr, const _Codecvt &__cvt, _State &__state, size_t &__count, _Fn __fn)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool std::\_\_str\_codecvt\_in (const char *__first, const char *__last, basic\_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool std::\_\_str\_codecvt\_in (const char *__first, const char *__last, basic\_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool std::\_\_str\_codecvt\_in\_all (const char *__first, const char *__last, basic\_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool std::\_\_str\_codecvt\_out (const _CharT *__first, const _CharT *__last, basic\_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool std::\_\_str\_codecvt\_out (const _CharT *__first, const _CharT *__last, basic\_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>`  
`bool std::\_\_str\_codecvt\_out\_all (const _CharT *__first, const _CharT *__last, basic\_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`

### 6.85.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.86 locale\_facets.h File Reference

### Classes

- class [std::\\_\\_ctype\\_abstract\\_base](#)< \_CharT >
- class [std::ctype](#)< \_CharT >
- class [std::ctype](#)< char >
- class [std::ctype](#)< wchar\_t >
- class [std::ctype\\_byname](#)< \_CharT >
- class [std::ctype\\_byname](#)< char >
- class [std::num\\_get](#)< \_CharT, \_InIter >
- class [std::num\\_put](#)< \_CharT, \_OutIter >
- class [std::num\\_punct](#)< \_CharT >
- class [std::num\\_punct\\_byname](#)< \_CharT >

### Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_NUM_CXX11_FACETS`
- `#define _GLIBCXX_NUM_FACETS`
- `#define _GLIBCXX_NUM_LBDL_ALT128_FACETS`
- `#define _GLIBCXX_NUM_UNICODE_FACETS`

## Functions

- `template<typename _CharT>`  
`_CharT * std::__add_grouping (_CharT * __s, _CharT __sep, const char * __gbeg, size_t __gsize, const _CharT`  
`* __first, const _CharT * __last)`
- `template<typename _Tp>`  
`void std::__convert_to_v (const char *, _Tp &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<> void std::__convert_to_v (const char *, double &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<> void std::__convert_to_v (const char *, float &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<> void std::__convert_to_v (const char *, long double &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT, typename _Outlter>`  
`_Outlter std::__write (_Outlter __s, const _CharT * __ws, int __len)`
- `template<typename _CharT>`  
`ostreambuf\_iterator< _CharT > std::__write (ostreambuf\_iterator< _CharT > __s, const _CharT * __ws, int`  
`__len)`
- `template<typename _CharT>`  
`bool std::isalnum (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::isalpha (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::isblank (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::iscntrl (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::isdigit (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::isgraph (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::islower (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::isprint (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::ispunct (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::isspace (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::isupper (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`bool std::isxdigit (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`_CharT std::tolower (_CharT __c, const locale & __loc)`
- `template<typename _CharT>`  
`_CharT std::toupper (_CharT __c, const locale & __loc)`

## Variables

- `template<typename _CharT>`  
`locale::id std::ctype< _CharT >::id`
- `template<typename _CharT, typename _InIter>`  
`locale::id std::num_get< _CharT, _InIter >::id`
- `template<typename _CharT, typename _OutIter>`  
`locale::id std::num_put< _CharT, _OutIter >::id`
- `template<typename _CharT>`  
`locale::id std::numpunct< _CharT >::id`

### 6.86.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.87 locale\_facets.tcc File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _LOCALE_FACETS_TCC`

### Functions

- `template<typename _CharT>`  
`_CharT * std::__add_grouping (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _CharT, typename _ValueT>`  
`int std::__int_to_char (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __dec)`
- `bool std::__verify_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`

### 6.87.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.88 locale\_facets\_nonio.h File Reference

### Classes

- class `std::messages< _CharT >`
- struct `std::messages_base`
- class `std::messages_byname< _CharT >`
- class `std::money_base`
- class `std::money_get< _CharT, _InIter >`
- class `std::money_put< _CharT, _OutIter >`
- class `std::moneypunct< _CharT, _Intl >`
- class `std::moneypunct_byname< _CharT, _Intl >`
- class `std::time_base`



- class `std::time_get<_CharT, _InIter>`
- class `std::time_get_byname<_CharT, _InIter>`
- class `std::time_put<_CharT, _OutIter>`
- class `std::time_put_byname<_CharT, _OutIter>`

## Namespaces

- namespace `std`

## Variables

- template<typename \_CharT>  
`locale::id std::__timepunct<_CharT>::id`
- template<typename \_CharT>  
const \_CharT \* `std::__timepunct_cache<_CharT>::S_timezones` [14]
- template<> const char \* `std::__timepunct_cache<char>::S_timezones` [14]
- template<> const wchar\_t \* `std::__timepunct_cache<wchar_t>::S_timezones` [14]
- template<typename \_CharT>  
`locale::id std::messages<_CharT>::id`
- template<typename \_CharT, typename \_InIter>  
`locale::id std::money_get<_CharT, _InIter>::id`
- template<typename \_CharT, typename \_OutIter>  
`locale::id std::money_put<_CharT, _OutIter>::id`
- template<typename \_CharT, bool \_Intl>  
`locale::id std::moneypunct<_CharT, _Intl>::id`
- template<typename \_CharT, bool \_Intl>  
const bool `std::moneypunct<_CharT, _Intl>::intl`
- template<typename \_CharT, bool \_Intl>  
const bool `std::moneypunct_byname<_CharT, _Intl>::intl`
- template<typename \_CharT, typename \_InIter>  
`locale::id std::time_get<_CharT, _InIter>::id`
- template<typename \_CharT, typename \_OutIter>  
`locale::id std::time_put<_CharT, _OutIter>::id`

### 6.88.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.89 locale\_facets\_nonio.tcc File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _LOCALE_FACETS_NONIO_TCC`

### 6.89.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.90 localefwd.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Facet>`  
`bool std::has\_facet (const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isblank (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet>`  
`const _Facet & std::use\_facet (const locale &__loc)`

#### 6.90.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.91 mask\_array.h File Reference

### Classes

- class [std::mask\\_array<\\_Tp>](#)

### Namespaces

- namespace [std](#)

### 6.91.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.92 `max_size_type.h` File Reference

### 6.92.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.93 `memory_resource.h` File Reference

### Classes

- struct `std::allocator_traits< pmr::polymorphic_allocator< _Tp > >`
- class `std::pmr::memory_resource`
- class `std::pmr::polymorphic_allocator< _Tp >`

### Namespaces

- namespace `std`

### Functions

- bool `std::pmr::operator==` (const `memory_resource` &\_\_a, const `memory_resource` &\_\_b) noexcept
- template<typename \_Tp1, typename \_Tp2>  
bool `std::pmr::operator==` (const `polymorphic_allocator`< \_Tp1 > &\_\_a, const `polymorphic_allocator`< \_Tp2 > &\_\_b) noexcept

### 6.93.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory_resource>`.

## 6.94 `memoryfwd.h` File Reference

### Namespaces

- namespace `std`

### 6.94.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.95 `mofunc_impl.h` File Reference

### Classes

- class `std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`

### Namespaces

- namespace `std`

## Macros

- `#define _GLIBCXX_MOF_CV_REF`
- `#define _GLIBCXX_MOF_INV_QUALS`
- `#define _GLIBCXX_MOF_REF`

### 6.95.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.96 monostate.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- constexpr strong\_ordering **std::operator<=>** (monostate, monostate) noexcept
- constexpr bool **std::operator==** (monostate, monostate) noexcept

### 6.96.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

## 6.97 move.h File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_FORWARD(_Tp, __val)`
- `#define _GLIBCXX_FWDREF(_Tp)`
- `#define _GLIBCXX_MOVE(__val)`

## Typedefs

- `template<typename _Tp, typename _Up>`  
`using std::__like_t`

## Functions

- `template<typename _Tp>`  
`constexpr _Tp * std::__addressof (_Tp &__r) noexcept`
- `template<typename _Tp, typename _Up = _Tp>`  
`constexpr _Tp std::__exchange (_Tp &__obj, _Up &&__new_val)`
- `template<typename _Tp>`  
`constexpr _Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp>`  
`const _Tp * std::addressof (const _Tp &&)=delete`
- `template<typename _Tp>`  
`constexpr _Tp && std::forward (typename std::remove_reference<_Tp>::type &&__t) noexcept`

- `template<typename _Tp>`  
`constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp>`  
`constexpr std::remove_reference< _Tp >::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp>`  
`constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_noexcept`  
`( _Tp &__x) noexcept`
- `template<typename _Tp>`  
`requires (! __is_tuple_like< _Tp >::value) && is_move_constructible_v< _Tp > && is_move_assignable_v< _Tp >`  
`constexpr void std::swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */) is_nothrow_move_assignable< _Tp`  
`> >`
- `template<typename _Tp, size_t _Nm>`  
`requires is_swappable_v< _Tp >`  
`constexpr void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(/*conditional */)`

### 6.97.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

## 6.98 nested\_exception.h File Reference

### Classes

- class `std::nested_exception`

### Namespaces

- namespace `std`

### Functions

- `template<typename _Ex>`  
`void std::rethrow_if_nested (const _Ex &__ex)`
- `template<typename _Tp>`  
`void std::throw_with_nested (_Tp &&__t)`

### 6.98.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 6.99 bits/new\_allocator.h File Reference

### Classes

- class `std::__new_allocator< _Tp >`

### Namespaces

- namespace `std`

## Macros

- `#define _GLIBCXX_OPERATOR_DELETE`
- `#define _GLIBCXX_OPERATOR_NEW`
- `#define _GLIBCXX_SIZED_DEALLOC(p, n)`

### 6.99.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.100 ext/new\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::new\\_allocator<\\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.100.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.101 node\_handle.h File Reference

### 6.101.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map,set,unordered_map,unordered_set>`.

## 6.102 ostream.h File Reference

### Classes

- class [std::basic\\_ostream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry](#)

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Os, typename _Tp>`  
using `std::__rvalue_stream_insertion_t`

### Functions

- `template<typename _Ostream, typename _Tp>`  
`__rvalue_stream_insertion_t<_Ostream, _Tp>` [std::operator<<](#) (`_Ostream &&__os`, `const _Tp &__x`)
- `template<typename _CharT, typename _Traits>`  
[basic\\_ostream<\\_CharT, \\_Traits>](#) & [std::operator<<](#) (`basic_ostream<_CharT, _Traits> &__out`, `_CharT __c`)

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, char16_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, char32_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, wchar_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, char16_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, char32_t)=delete`
  
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, const char16_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, const char32_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, const wchar_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, const char16_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, const char32_t *)=delete`

### 6.102.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 6.103 ostream.tcc File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _OSTREAM_TCC`

### Functions

- `template<typename _CharT, typename _Traits>  
basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__out, const char  
*__s)`

#### 6.103.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 6.104 ostream\_insert.h File Reference

### Namespaces

- namespace [std](#)

#### 6.104.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 6.105 out\_ptr.h File Reference

#### 6.105.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.106 parse\_numbers.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<unsigned long long _Val>  
using std::\_\_parse\_int::\_\_ull\_constant`
- `template<char... _Digs>  
using std::\_\_select\_int::\_\_Select\_int`

#### 6.106.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<chrono>`.



## 6.107 postypes.h File Reference

### Classes

- class [std::fpos< \\_StateT >](#)

### Namespaces

- namespace [std](#)

### Typedefs

- typedef long long [std::streamoff](#)
- typedef [fpos< mbstate\\_t >](#) [std::streampos](#)
- typedef ptrdiff\_t [std::streamsize](#)
- typedef [fpos< mbstate\\_t >](#) [std::u16streampos](#)
- typedef [fpos< mbstate\\_t >](#) [std::u32streampos](#)
- typedef [fpos< mbstate\\_t >](#) [std::wstreampos](#)

### Functions

- template<typename \_StateT>  
bool **std::operator!=** (const [fpos< \\_StateT >](#) &\_\_lhs, const [fpos< \\_StateT >](#) &\_\_rhs)
- template<typename \_StateT>  
bool **std::operator==** (const [fpos< \\_StateT >](#) &\_\_lhs, const [fpos< \\_StateT >](#) &\_\_rhs)

#### 6.107.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 6.108 predefined\_ops.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- template<typename \_Compare>  
constexpr [\\_Iter\\_comp\\_iter< \\_Compare >](#) [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_iter](#) (\_Compare \_\_comp)
- template<typename \_Compare, typename \_Iterator>  
constexpr [\\_Iter\\_comp\\_to\\_iter< \\_Compare, \\_Iterator >](#) [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_iter](#) (\_Iter\_comp\_↵  
[\\_iter< \\_Compare >](#) \_\_comp, \_Iterator \_\_it)
- template<typename \_Iterator>  
constexpr [\\_Iter\\_equals\\_iter< \\_Iterator >](#) [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_iter](#) (\_Iter\_equal\_to\_iter, \_Iterator  
\_\_it)
- template<typename \_Compare>  
constexpr [\\_Iter\\_comp\\_val< \\_Compare >](#) [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_val](#) (\_Compare \_\_comp)
- template<typename \_Compare, typename \_Value>  
[\\_Iter\\_comp\\_to\\_val< \\_Compare, \\_Value >](#) constexpr [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_val](#) (\_Compare \_\_↵  
comp, \_Value &\_\_val)
- template<typename \_Compare>  
constexpr [\\_Iter\\_comp\\_val< \\_Compare >](#) [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_val](#) (\_Iter\_comp\_iter< \_Compare  
> \_\_comp)

- constexpr `_Iter_equal_to_val` `__gnu_cxx::__ops::__iter_comp_val` (`_Iter_equal_to_iter`)
- constexpr `_Iter_less_val` `__gnu_cxx::__ops::__iter_comp_val` (`_Iter_less_iter`)
- constexpr `_Iter_equal_to_iter` `__gnu_cxx::__ops::__iter_equal_to_iter` ()
- constexpr `_Iter_equal_to_val` `__gnu_cxx::__ops::__iter_equal_to_val` ()
- template<typename `_Value`>  
constexpr `_Iter_equals_val`< `_Value` > `__gnu_cxx::__ops::__iter_equals_val` (`_Value` & `__val`)
- constexpr `_Iter_less_iter` `__gnu_cxx::__ops::__iter_less_iter` ()
- constexpr `_Iter_less_val` `__gnu_cxx::__ops::__iter_less_val` ()
- template<typename `_Predicate`>  
constexpr `_Iter_negate`< `_Predicate` > `__gnu_cxx::__ops::__negate` (`_Iter_pred`< `_Predicate` > `__pred`)
- template<typename `_Predicate`>  
constexpr `_Iter_pred`< `_Predicate` > `__gnu_cxx::__ops::__pred_iter` (`_Predicate` `__pred`)
- template<typename `_Compare`>  
constexpr `_Val_comp_iter`< `_Compare` > `__gnu_cxx::__ops::__val_comp_iter` (`_Compare` `__comp`)
- template<typename `_Compare`>  
constexpr `_Val_comp_iter`< `_Compare` > `__gnu_cxx::__ops::__val_comp_iter` (`_Iter_comp_iter`< `_Compare` > `__comp`)
- constexpr `_Val_less_iter` `__gnu_cxx::__ops::__val_comp_iter` (`_Iter_less_iter`)
- constexpr `_Val_less_iter` `__gnu_cxx::__ops::__val_less_iter` ()

### 6.108.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly. Instead, include `<algorithm>`.

## 6.109 ptr\_traits.h File Reference

### Classes

- struct `std::pointer_traits<_Ptr>`
- struct `std::pointer_traits<_Tp*>`

### Namespaces

- namespace `__gnu_debug`
- namespace `std`

### Typedefs

- template<typename `_Ptr`, typename `_Tp`>  
using `std::__ptr_rebind`

### Functions

- template<typename `_Tp`>  
constexpr `_Tp*` `std::to_address` (`_Tp*` `__ptr`) noexcept
- template<typename `_Ptr`>  
constexpr auto `std::to_address` (const `_Ptr` & `__ptr`) noexcept

### 6.109.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.110 `quoted_string.h` File Reference

### Classes

- struct `std::__detail::Quoted_string<_String, _CharT>`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Functions

- template<typename \_CharT, typename \_Traits, typename \_String>  
`std::basic_ostream<_CharT, _Traits> & std::__detail::operator<< (std::basic_ostream<_CharT, _Traits> &↔  
__os, const Quoted_string<_String, _CharT> &__str)`
- template<typename \_CharT, typename \_Traits>  
`std::basic_ostream<_CharT, _Traits> & std::__detail::operator<< (std::basic_ostream<_CharT, _Traits> &↔  
__os, const Quoted_string<const _CharT*, _CharT> &__str)`
- template<typename \_CharT, typename \_Traits, typename \_Alloc>  
`std::basic_istream<_CharT, _Traits> & std::__detail::operator>> (std::basic_istream<_CharT, _Traits> &↔  
__is, const Quoted_string<basic_string<_CharT, _Traits, _Alloc> &, _CharT> &__str)`

#### 6.110.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iomanip>`.

## 6.111 `random.h` File Reference

### Classes

- class `std::bernoulli_distribution`
- class `std::binomial_distribution<_IntType>`
- class `std::cauchy_distribution<_RealType>`
- class `std::chi_squared_distribution<_RealType>`
- class `std::discard_block_engine<_RandomNumberEngine, __p, __r>`
- class `std::discrete_distribution<_IntType>`
- class `std::exponential_distribution<_RealType>`
- class `std::extreme_value_distribution<_RealType>`
- class `std::fisher_f_distribution<_RealType>`
- class `std::gamma_distribution<_RealType>`
- class `std::geometric_distribution<_IntType>`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>`
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m>`
- class `std::lognormal_distribution<_RealType>`
- class `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>`
- class `std::negative_binomial_distribution<_IntType>`
- class `std::normal_distribution<_RealType>`
- struct `std::uniform_real_distribution<_RealType>::param_type`
- struct `std::normal_distribution<_RealType>::param_type`
- struct `std::lognormal_distribution<_RealType>::param_type`
- struct `std::gamma_distribution<_RealType>::param_type`
- struct `std::chi_squared_distribution<_RealType>::param_type`

- struct `std::cauchy_distribution<_RealType>::param_type`
- struct `std::fisher_f_distribution<_RealType>::param_type`
- struct `std::student_t_distribution<_RealType>::param_type`
- struct `std::bernoulli_distribution::param_type`
- struct `std::binomial_distribution<_IntType>::param_type`
- struct `std::geometric_distribution<_IntType>::param_type`
- struct `std::negative_binomial_distribution<_IntType>::param_type`
- struct `std::poisson_distribution<_IntType>::param_type`
- struct `std::exponential_distribution<_RealType>::param_type`
- struct `std::weibull_distribution<_RealType>::param_type`
- struct `std::extreme_value_distribution<_RealType>::param_type`
- struct `std::discrete_distribution<_IntType>::param_type`
- struct `std::piecewise_constant_distribution<_RealType>::param_type`
- struct `std::piecewise_linear_distribution<_RealType>::param_type`
- class `std::piecewise_constant_distribution<_RealType>`
- class `std::piecewise_linear_distribution<_RealType>`
- class `std::poisson_distribution<_IntType>`
- class `std::random_device`
- class `std::seed_seq`
- class `std::shuffle_order_engine<_RandomNumberEngine, __k>`
- class `std::student_t_distribution<_RealType>`
- class `std::subtract_with_carry_engine<_UIntType, __w, __s, __r>`
- class `std::uniform_real_distribution<_RealType>`
- class `std::weibull_distribution<_RealType>`

## Namespaces

- namespace `std`

## Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` **`std::minstd_rand`**
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >` **`std::minstd_rand0`**
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >` **`std::mt19937`**
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL >` **`std::mt19937_64`**
- typedef `discard_block_engine< ranlux24_base, 223, 23 >` **`std::ranlux24`**
- typedef `subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >` **`std::ranlux24_base`**
- typedef `discard_block_engine< ranlux48_base, 389, 11 >` **`std::ranlux48`**
- typedef `subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 >` **`std::ranlux48_base`**

## Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>`  
`_RealType std::generate_canonical(_UniformRandomNumberGenerator &__g)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_real_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::bernoulli_distribution &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &`  
`std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &`  
`std::uniform_real_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::bernoulli_distribution &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::geometric_distribution< _IntType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::weibull_distribution< _RealType > &__x)`

### 6.111.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 6.112 bits/random.tcc File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Macros

- `#define _RANDOM_TCC`

### Functions

- `template<typename _ValT, typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::__detail::__extract_params (basic_istream< _CharT, _Traits > &__is,`  
`vector< _ValT > &__vals, size_t __n)`
- `template<typename _Tp>`  
`constexpr bool std::__detail::__p1_representable_as_double (_Tp __x) noexcept`
- `template<typename _Tp>`  
`constexpr bool std::__detail::__representable_as_double (_Tp __x) noexcept`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>`  
`_RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_real_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const chi_squared_distribution< _RealType > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const gamma_distribution< _RealType > &__x)`

- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const lognormal_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f`  
`> &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const negative_binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const poisson_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::bernoulli_distribution &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const student_t_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`

- `template<typename _RealType>`  
`bool std::operator== (const std::normal\_distribution< _RealType > &__d1, const std::normal\_distribution< _RealType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &, std::uniform\_int\_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &, std::uniform\_real\_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, binomial\_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, chi\_squared\_distribution< _RealType > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, discard\_block\_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, discrete\_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, fisher\_f\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, gamma\_distribution< _RealType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, linear\_congruential\_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, lognormal\_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, negative\_binomial\_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, normal\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, piecewise\_constant\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, piecewise\_linear\_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__, poisson\_distribution< _IntType > &__x)`



- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`

### 6.112.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 6.113 ext/random.tcc File Reference

### Namespaces

- namespace `__gnu_cxx`

### Macros

- `#define _EXT_RANDOM_TCC`

### Functions

- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<typename _UIntType, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>`

- ```
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits >
& __os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1,
__sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > & __x)
```
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const \_\_gnu\_cxx::triangular\_distribution< \_RealType > & \_\_x)
  - template<std::size\_t \_Dimen, typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const \_\_gnu\_cxx::uniform\_inside\_sphere\_distribution< \_Dimen, \_RealType > & \_\_x)
  - template<std::size\_t \_Dimen, typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const \_\_gnu\_cxx::uniform\_on\_sphere\_distribution< \_Dimen, \_RealType > & \_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const \_\_gnu\_cxx::von\_mises\_distribution< \_RealType > & \_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const arcsine\_distribution< \_RealType > & \_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const hoyt\_distribution< \_RealType > & \_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const k\_distribution< \_RealType > & \_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const logistic\_distribution< \_RealType > & \_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const nakagami\_distribution< \_RealType > & \_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const pareto\_distribution< \_RealType > & \_\_x)
  - template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< \_CharT, \_Traits > & \_\_os, const rice\_distribution< \_RealType > & \_\_x)
  - template<size\_t \_Dimen, typename \_RealType>  
bool \_\_gnu\_cxx::operator== (const \_\_gnu\_cxx::normal\_mv\_distribution< \_Dimen, \_RealType > & \_\_d1, const \_\_gnu\_cxx::normal\_mv\_distribution< \_Dimen, \_RealType > & \_\_d2)
  - template<typename \_UIntType, size\_t \_\_m, size\_t \_\_pos1, size\_t \_\_sl1, size\_t \_\_sl2, size\_t \_\_sr1, size\_t \_\_sr2, uint32\_t \_\_msk1, uint32\_t \_\_msk2, uint32\_t \_\_msk3, uint32\_t \_\_msk4, uint32\_t \_\_parity1, uint32\_t \_\_parity2, uint32\_t \_\_parity3, uint32\_t \_\_parity4>  
bool \_\_gnu\_cxx::operator== (const \_\_gnu\_cxx::simd\_fast\_mersenne\_twister\_engine< \_UIntType, \_\_m, \_\_pos1, \_\_sl1, \_\_sl2, \_\_sr1, \_\_sr2, \_\_msk1, \_\_msk2, \_\_msk3, \_\_msk4, \_\_parity1, \_\_parity2, \_\_parity3, \_\_parity4 > & \_\_lhs, const \_\_gnu\_cxx::simd\_fast\_mersenne\_twister\_engine< \_UIntType, \_\_m, \_\_pos1, \_\_sl1, \_\_sl2, \_\_sr1, \_\_sr2, \_\_msk1, \_\_msk2, \_\_msk3, \_\_msk4, \_\_parity1, \_\_parity2, \_\_parity3, \_\_parity4 > & \_\_rhs)
  - template<typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_istream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator>> (std::basic\_istream< \_CharT, \_Traits > & \_\_is, \_\_gnu\_cxx::beta\_distribution< \_RealType > & \_\_x)
  - template<typename \_UIntType, typename \_CharT, typename \_Traits>  
std::basic\_istream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator>> (std::basic\_istream< \_CharT, \_Traits > & \_\_is, \_\_gnu\_cxx::hypergeometric\_distribution< \_UIntType > & \_\_x)
  - template<size\_t \_Dimen, typename \_RealType, typename \_CharT, typename \_Traits>  
std::basic\_istream< \_CharT, \_Traits > & \_\_gnu\_cxx::operator>> (std::basic\_istream< \_CharT, \_Traits > & \_\_is, \_\_gnu\_cxx::normal\_mv\_distribution< \_Dimen, \_RealType > & \_\_x)

- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::triangular_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::von_mises_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, k_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, logistic_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, nakagami_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &__is, rice_distribution< _RealType > &__x)`

### 6.113.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/random>`.

## 6.114 range\_access.h File Reference

### Namespaces

- namespace `std`

### Functions

- `template<typename _Container>`  
`constexpr auto std::begin(_Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())`

- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * std::begin (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::begin (const _Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.  
begin())`
- `template<class _Tp>`  
`const _Tp * std::begin (const valarray< _Tp > &__va) noexcept`
- `template<class _Tp>`  
`_Tp * std::begin (valarray< _Tp > &__va) noexcept`
- `template<typename _Container>`  
`constexpr auto std::cbegin (const _Container &__cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(  
__cont))`
- `template<typename _Container>`  
`constexpr auto std::cend (const _Container &__cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(  
__cont))`
- `template<typename _Container>`  
`constexpr auto std::crbegin (const _Container &__cont) noexcept(noexcept(std::rbegin(__cont))) -> decltype(std::rbegin(  
__cont))`
- `template<typename _Container>`  
`constexpr auto std::crend (const _Container &__cont) noexcept(noexcept(std::rend(__cont))) -> decltype(std::rend(  
__cont))`
- `template<typename _Container>`  
`constexpr auto std::data (_Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * std::data (_Tp(&__array)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::data (const _Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.  
data())`
- `template<typename _Tp>`  
`constexpr const _Tp * std::data (initializer\_list< _Tp > __il) noexcept`
- `template<typename _Container>`  
`constexpr auto std::empty (const _Container &__cont) noexcept(noexcept(__cont.empty())) -> decltype(__  
cont.empty())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr bool std::empty (const _Tp(&)[_Nm]) noexcept`
- `template<typename _Tp>`  
`constexpr bool std::empty (initializer\_list< _Tp > __il) noexcept`
- `template<typename _Container>`  
`constexpr auto std::end (_Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * std::end (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::end (const _Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())`
- `template<class _Tp>`  
`const _Tp * std::end (const valarray< _Tp > &__va) noexcept`
- `template<class _Tp>`  
`_Tp * std::end (valarray< _Tp > &__va) noexcept`
- `template<typename _Container>`  
`constexpr auto std::rbegin (_Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.  
rbegin())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse\_iterator< _Tp * > std::rbegin (_Tp(&__arr)[_Nm]) noexcept`

- `template<typename _Container>`  
`constexpr auto std::rbegin (const _Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.rbegin())`
- `template<typename _Tp>`  
`constexpr reverse\_iterator< const _Tp * > std::rbegin (initializer_list< _Tp > __il) noexcept`
- `template<typename _Container>`  
`constexpr auto std::rend (_Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse\_iterator< _Tp * > std::rend (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::rend (const _Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())`
- `template<typename _Tp>`  
`constexpr reverse\_iterator< const _Tp * > std::rend (initializer_list< _Tp > __il) noexcept`
- `template<typename _Container>`  
`constexpr auto std::size (const _Container &__cont) noexcept(noexcept(__cont.size())) -> decltype(__cont.size())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr size_t std::size (const _Tp(&)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::ssize (const _Container &__cont) noexcept(noexcept(__cont.size())) -> common\_type\_t< ptrdiff_t, make\_signed\_t< decltype(__cont.size())> >`
- `template<typename _Tp, ptrdiff_t _Num>`  
`constexpr ptrdiff_t std::ssize (const _Tp(&)[_Num]) noexcept`

#### 6.114.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.115 `ranges_algo.h` File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Fp>`  
`using std::ranges::\_\_detail::\_\_by\_ref\_or\_value\_fn`
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
`using std::ranges::binary\_transform\_result`
- `template<typename _Iter, typename _Out>`  
`using std::ranges::copy\_if\_result`
- `template<typename _Iter, typename _Fp>`  
`using std::ranges::for\_each\_n\_result`
- `template<typename _Iter, typename _Fp>`  
`using std::ranges::for\_each\_result`
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
`using std::ranges::merge\_result`
- `template<typename _Iter>`  
`using std::ranges::minmax\_element\_result`
- `template<typename _Tp>`  
`using std::ranges::minmax\_result`

- `template<typename _Iter>`  
using **std::ranges::next\_permutation\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::partial\_sort\_copy\_result**
- `template<typename _Iter, typename _Out1, typename _Out2>`  
using **std::ranges::partition\_copy\_result**
- `template<typename _Iter>`  
using **std::ranges::prev\_permutation\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::remove\_copy\_if\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::remove\_copy\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::replace\_copy\_if\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::replace\_copy\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::reverse\_copy\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::rotate\_copy\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::set\_difference\_result**
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
using **std::ranges::set\_intersection\_result**
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
using **std::ranges::set\_symmetric\_difference\_result**
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
using **std::ranges::set\_union\_result**
- `template<typename _Iter1, typename _Iter2>`  
using **std::ranges::swap\_ranges\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::unary\_transform\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::unique\_copy\_result**

## Functions

- `template<typename _Iter, typename _Comp>`  
constexpr void **std::ranges::\_\_detail::\_\_adjust\_heap** (\_Iter \_\_first, iter\_difference\_t< \_Iter > \_\_holeIndex, iter\_difference\_t< \_Iter > \_\_len, iter\_value\_t< \_Iter > \_\_value, \_Comp \_\_comp)
- `template<typename _Iter, typename _Distance, typename _Compare>`  
constexpr void **std::ranges::\_\_detail::\_\_chunk\_insertion\_sort** (\_Iter \_\_first, \_Iter \_\_last, \_Distance \_\_chunk\_size, \_Compare \_\_comp)
- `template<typename _Iter, typename _Comp>`  
constexpr void **std::ranges::\_\_detail::\_\_final\_insertion\_sort** (\_Iter \_\_first, \_Iter \_\_last, \_Comp \_\_comp)
- `template<typename _Iter, typename _Pred, typename _Distance>`  
constexpr \_Iter **std::ranges::\_\_detail::\_\_find\_if\_not\_n** (\_Iter \_\_first, \_Distance & \_\_len, \_Pred \_\_pred)
- `template<typename _Iter, typename _Comp>`  
constexpr void **std::ranges::\_\_detail::\_\_heap\_select** (\_Iter \_\_first, \_Iter \_\_middle, \_Iter \_\_last, \_Comp \_\_comp)
- `template<typename _Iter, typename _Comp>`  
constexpr void **std::ranges::\_\_detail::\_\_inplace\_stable\_sort** (\_Iter \_\_first, \_Iter \_\_last, \_Comp \_\_comp)
- `template<typename _Iter, typename _Comp>`  
constexpr void **std::ranges::\_\_detail::\_\_insertion\_sort** (\_Iter \_\_first, \_Iter \_\_last, \_Comp \_\_comp)

- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::detail::__introsselect (_Iter __first, _Iter __nth, _Iter __last, iter_difference_t<_Iter> __depth_limit, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::detail::__introsort_loop (_Iter __first, _Iter __last, unsigned __depth_limit, _Comp __comp)`
- `template<typename _Comp, typename _Proj>`  
`constexpr _Comp_proj<_Comp, _Proj> std::ranges::detail::__make_comp_proj (_Comp &__comp, _Proj &__proj)`
- `template<typename _Pred, typename _Proj>`  
`constexpr _Pred_proj<_Pred, _Proj> std::ranges::detail::__make_pred_proj (_Pred &__pred, _Proj &__proj)`
- `template<typename _Iter, typename _Pointer, typename _Comp>`  
`void std::ranges::detail::__merge_adaptive (_Iter __first, _Iter __middle, _Iter __last, iter_difference_t<_Iter> __len1, iter_difference_t<_Iter> __len2, _Pointer __buffer, _Comp __comp)`
- `template<typename _Iter, typename _Distance, typename _Pointer, typename _Comp>`  
`void std::ranges::detail::__merge_adaptive_resize (_Iter __first, _Iter __middle, _Iter __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Comp __comp)`
- `template<typename _Iter, typename _Out, typename _Distance, typename _Comp>`  
`void std::ranges::detail::__merge_sort_loop (_Iter __first, _Iter __last, _Out __result, _Distance __step_size, _Comp __comp)`
- `template<typename _Iter, typename _Pointer, typename _Comp>`  
`void std::ranges::detail::__merge_sort_with_buffer (_Iter __first, _Iter __last, _Pointer __buffer, _Comp __comp)`
- `template<typename _Iter, typename _Distance, typename _Comp>`  
`constexpr void std::ranges::detail::__merge_without_buffer (_Iter __first, _Iter __middle, _Iter __last, _Distance __len1, _Distance __len2, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::detail::__move_median_to_first (_Iter __result, _Iter __a, _Iter __b, _Iter __c, _Comp __comp)`
- `template<typename _Iter, typename _Out, typename _Comp>`  
`_Out std::ranges::detail::__move_merge (_Iter __first1, _Iter __last1, _Iter __first2, _Iter __last2, _Out __result, _Comp __comp)`
- `template<typename _Iter1, typename _Iter2, typename _Out, typename _Comp>`  
`void std::ranges::detail::__move_merge_adaptive (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 __last2, _Out __result, _Comp __comp)`
- `template<typename _Iter1, typename _Iter2, typename _Iter3, typename _Comp>`  
`void std::ranges::detail::__move_merge_adaptive_backward (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 __last2, _Iter3 __result, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::detail::__partial_sort (_Iter __first, _Iter __middle, _Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::detail::__pop_heap (_Iter __first, _Iter __last, _Iter __result, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::detail::__push_heap (_Iter __first, iter_difference_t<_Iter> __holeIndex, iter_difference_t<_Iter> __topIndex, iter_value_t<_Iter> __value, _Comp __comp)`
- `template<typename _Iter1, typename _Iter2>`  
`_Iter1 std::ranges::detail::__rotate_adaptive (_Iter1 __first, _Iter1 __middle, _Iter1 __last, iter_difference_t<_Iter1> __len1, iter_difference_t<_Iter1> __len2, _Iter2 __buffer, iter_difference_t<_Iter1> __buffer_size)`
- `template<typename _Iter, typename _Sent, typename _Pointer, typename _Pred, typename _Distance>`  
`constexpr subrange<_Iter> std::ranges::detail::__stable_partition_adaptive (_Iter __first, _Sent __last, _Pred __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`

- `template<typename _Iter, typename _Pointer, typename _Comp>`  
`void std::ranges::__detail::__stable_sort_adaptive (_Iter __first, _Iter __middle, _Iter __last, _Pointer __buffer, _Comp __comp)`
- `template<typename _Iter, typename _Pointer, typename _Distance, typename _Comp>`  
`void std::ranges::__detail::__stable_sort_adaptive_resize (_Iter __first, _Iter __last, _Pointer __buffer, _Distance __buffer_size, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__unguarded_insertion_sort (_Iter __first, _Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__unguarded_linear_insert (_Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr _Iter std::ranges::__detail::__unguarded_partition (_Iter __first, _Iter __last, _Iter __pivot, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr _Iter std::ranges::__detail::__unguarded_partition_pivot (_Iter __first, _Iter __last, _Comp __comp)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::shift_left (_ForwardIterator __first, _ForwardIterator __last, typename iterator\_traits<_ForwardIterator>::difference_type __n)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::shift_right (_ForwardIterator __first, _ForwardIterator __last, typename iterator\_traits<_ForwardIterator>::difference_type __n)`

## Variables

- `constexpr int std::ranges::__detail::__sort_threshold`
- `constexpr __all_of_fn std::ranges::all_of`
- `constexpr __any_of_fn std::ranges::any_of`
- `constexpr __binary_search_fn std::ranges::binary_search`
- `constexpr __clamp_fn std::ranges::clamp`
- `constexpr __copy_if_fn std::ranges::copy_if`
- `constexpr __count_fn std::ranges::count`
- `constexpr __count_if_fn std::ranges::count_if`
- `constexpr __equal_range_fn std::ranges::equal_range`
- `constexpr __find_end_fn std::ranges::find_end`
- `constexpr __find_first_of_fn std::ranges::find_first_of`
- `constexpr __for_each_fn std::ranges::for_each`
- `constexpr __for_each_n_fn std::ranges::for_each_n`
- `constexpr __generate_fn std::ranges::generate`
- `constexpr __generate_n_fn std::ranges::generate_n`
- `constexpr __includes_fn std::ranges::includes`
- `constexpr __inplace_merge_fn std::ranges::inplace_merge`
- `constexpr __is_heap_fn std::ranges::is_heap`
- `constexpr __is_heap_until_fn std::ranges::is_heap_until`
- `constexpr __is_partitioned_fn std::ranges::is_partitioned`
- `constexpr __is_permutation_fn std::ranges::is_permutation`
- `constexpr __is_sorted_fn std::ranges::is_sorted`
- `constexpr __is_sorted_until_fn std::ranges::is_sorted_until`
- `constexpr __lexicographical_compare_fn std::ranges::lexicographical_compare`
- `constexpr __lower_bound_fn std::ranges::lower_bound`
- `constexpr __make_heap_fn std::ranges::make_heap`
- `constexpr __max_fn std::ranges::max`



- constexpr \_\_max\_element\_fn **std::ranges::max\_element**
- constexpr \_\_merge\_fn **std::ranges::merge**
- constexpr \_\_min\_element\_fn **std::ranges::min\_element**
- constexpr \_\_minmax\_fn **std::ranges::minmax**
- constexpr \_\_minmax\_element\_fn **std::ranges::minmax\_element**
- constexpr \_\_next\_permutation\_fn **std::ranges::next\_permutation**
- constexpr \_\_none\_of\_fn **std::ranges::none\_of**
- constexpr \_\_nth\_element\_fn **std::ranges::nth\_element**
- constexpr \_\_partial\_sort\_fn **std::ranges::partial\_sort**
- constexpr \_\_partial\_sort\_copy\_fn **std::ranges::partial\_sort\_copy**
- constexpr \_\_partition\_fn **std::ranges::partition**
- constexpr \_\_partition\_copy\_fn **std::ranges::partition\_copy**
- constexpr \_\_partition\_point\_fn **std::ranges::partition\_point**
- constexpr \_\_pop\_heap\_fn **std::ranges::pop\_heap**
- constexpr \_\_prev\_permutation\_fn **std::ranges::prev\_permutation**
- constexpr \_\_push\_heap\_fn **std::ranges::push\_heap**
- constexpr \_\_remove\_fn **std::ranges::remove**
- constexpr \_\_remove\_copy\_fn **std::ranges::remove\_copy**
- constexpr \_\_remove\_copy\_if\_fn **std::ranges::remove\_copy\_if**
- constexpr \_\_remove\_if\_fn **std::ranges::remove\_if**
- constexpr \_\_replace\_fn **std::ranges::replace**
- constexpr \_\_replace\_copy\_fn **std::ranges::replace\_copy**
- constexpr \_\_replace\_copy\_if\_fn **std::ranges::replace\_copy\_if**
- constexpr \_\_replace\_if\_fn **std::ranges::replace\_if**
- constexpr \_\_reverse\_fn **std::ranges::reverse**
- constexpr \_\_reverse\_copy\_fn **std::ranges::reverse\_copy**
- constexpr \_\_rotate\_fn **std::ranges::rotate**
- constexpr \_\_rotate\_copy\_fn **std::ranges::rotate\_copy**
- constexpr \_\_sample\_fn **std::ranges::sample**
- constexpr \_\_search\_n\_fn **std::ranges::search\_n**
- constexpr \_\_set\_difference\_fn **std::ranges::set\_difference**
- constexpr \_\_set\_intersection\_fn **std::ranges::set\_intersection**
- constexpr \_\_set\_symmetric\_difference\_fn **std::ranges::set\_symmetric\_difference**
- constexpr \_\_set\_union\_fn **std::ranges::set\_union**
- constexpr \_\_shuffle\_fn **std::ranges::shuffle**
- constexpr \_\_sort\_fn **std::ranges::sort**
- constexpr \_\_sort\_heap\_fn **std::ranges::sort\_heap**
- constexpr \_\_stable\_partition\_fn **std::ranges::stable\_partition**
- constexpr \_\_stable\_sort\_fn **std::ranges::stable\_sort**
- constexpr \_\_swap\_ranges\_fn **std::ranges::swap\_ranges**
- constexpr \_\_transform\_fn **std::ranges::transform**
- constexpr \_\_unique\_fn **std::ranges::unique**
- constexpr \_\_unique\_copy\_fn **std::ranges::unique\_copy**
- constexpr \_\_upper\_bound\_fn **std::ranges::upper\_bound**

### 6.115.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 6.116 ranges\_algobase.h File Reference

### 6.116.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 6.117 ranges\_base.h File Reference

### Macros

- `#define _GLIBCXX26_RANGE_ALGO_DEF_VAL_T(_I, _P)`

### 6.117.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ranges>`.

## 6.118 ranges\_cmp.h File Reference

### Classes

- struct [std::ranges::equal\\_to](#)
- struct [std::ranges::greater](#)
- struct [std::ranges::greater\\_equal](#)
- struct [std::identity](#)
- struct [std::ranges::less](#)
- struct [std::ranges::less\\_equal](#)
- struct [std::ranges::not\\_equal\\_to](#)

### Namespaces

- namespace [std](#)

### 6.118.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.119 ranges\_uninitialized.h File Reference

### 6.119.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.120 ranges\_util.h File Reference

### Classes

- class [std::ranges::subrange<\\_It, \\_Sent, \\_Kind>](#)
- class [std::ranges::view\\_interface<\\_Derived>](#)

### Namespaces

- namespace [std](#)

## Typedefs

- `template<range _Range>`  
using **`std::ranges::borrowed_subrange_t`**
- `template<typename _Iter1, typename _Iter2>`  
using **`std::ranges::mismatch_result`**

## Enumerations

- enum class **`subrange_kind`** : `bool` { **`unsized`** , **`sized`** }

## Functions

- `template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>`  
requires (`(_Num == 0 && copyable<_It>) || _Num == 1`)  
constexpr auto **`std::get`** (const subrange< \_It, \_Sent, \_Kind > &\_\_r)
- `template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>`  
requires (`(_Num == 0 && copyable<_It>) || _Num == 1`)  
constexpr auto **`std::ranges::get`** (const [subrange](#)< \_It, \_Sent, \_Kind > &\_\_r)
- `template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>`  
requires (`_Num < 2`)  
constexpr auto **`std::ranges::get`** ([subrange](#)< \_It, \_Sent, \_Kind > &&\_\_r)
- `template<input_or_output_iterator _It, sentinel_for< _It > _Sent>`  
**`std::ranges::subrange`** (\_It, \_Sent) -> subrange< \_It, \_Sent >
- `template<input_or_output_iterator _It, sentinel_for< _It > _Sent>`  
**`std::ranges::subrange`** (\_It, \_Sent, \_\_detail::\_\_make\_unsigned\_like\_t< iter\_difference\_t< \_It > >) ->  
subrange< \_It, \_Sent, subrange\_kind::sized >
- `template<borrowed_range _Rng>`  
**`std::ranges::subrange`** (\_Rng &&) -> subrange< iterator\_t< \_Rng >, sentinel\_t< \_Rng >, (sized\_range< \_Rng > || sized\_sentinel\_for< sentinel\_t< \_Rng >, iterator\_t< \_Rng > >) ? subrange\_kind::sized : subrange\_kind::unsized >
- `template<borrowed_range _Rng>`  
**`std::ranges::subrange`** (\_Rng &&, \_\_detail::\_\_make\_unsigned\_like\_t< range\_difference\_t< \_Rng > >) ->  
subrange< iterator\_t< \_Rng >, sentinel\_t< \_Rng >, subrange\_kind::sized >

## Variables

- constexpr \_\_adjacent\_find\_fn **`std::ranges::adjacent_find`**
- `template<typename _It, typename _Sent, subrange_kind _Kind>`  
constexpr bool **`std::ranges::enable_borrowed_range`**< subrange< \_It, \_Sent, \_Kind > >
- constexpr \_\_find\_fn **`std::ranges::find`**
- constexpr \_\_find\_if\_fn **`std::ranges::find_if`**
- constexpr \_\_find\_if\_not\_fn **`std::ranges::find_if_not`**
- constexpr \_\_min\_fn **`std::ranges::min`**
- constexpr \_\_mismatch\_fn **`std::ranges::mismatch`**
- constexpr \_\_search\_fn **`std::ranges::search`**

### 6.120.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ranges>`.

## 6.121 refwrap.h File Reference

### Classes

- class [std::reference\\_wrapper<\\_Tp>](#)

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_Tp>  
**std::reference\_wrapper** (\_Tp &) -> [reference\\_wrapper<\\_Tp>](#)

#### 6.121.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.122 regex.h File Reference

### Classes

- class [std::basic\\_regex<\\_Ch\\_type, \\_Rx\\_traits>](#)
- class [std::match\\_results<\\_Bi\\_iter, \\_Alloc>](#)
- class [std::regex\\_iterator<\\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits>](#)
- class [std::regex\\_token\\_iterator<\\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits>](#)
- class [std::regex\\_traits<\\_Ch\\_type>](#)
- class [std::sub\\_match<\\_Biter>](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Typedefs

- typedef [match\\_results<const char\\*>](#) **std::cmatch**
- typedef [regex\\_iterator<const char\\*>](#) **std::cregex\_iterator**
- typedef [regex\\_token\\_iterator<const char\\*>](#) **std::cregex\_token\_iterator**
- typedef [sub\\_match<const char\\*>](#) **std::csub\_match**
- typedef [basic\\_regex<char>](#) **std::regex**
- typedef [match\\_results<string::const\\_iterator>](#) **std::smatch**
- typedef [regex\\_iterator<string::const\\_iterator>](#) **std::sregex\_iterator**
- typedef [regex\\_token\\_iterator<string::const\\_iterator>](#) **std::sregex\_token\_iterator**
- typedef [sub\\_match<string::const\\_iterator>](#) **std::ssub\_match**
- typedef [match\\_results<const wchar\\_t\\*>](#) **std::wcmatch**
- typedef [regex\\_iterator<const wchar\\_t\\*>](#) **std::wcregex\_iterator**
- typedef [regex\\_token\\_iterator<const wchar\\_t\\*>](#) **std::wcregex\_token\_iterator**
- typedef [sub\\_match<const wchar\\_t\\*>](#) **std::wcs\_sub\_match**
- typedef [basic\\_regex<wchar\\_t>](#) **std::wregex**
- typedef [match\\_results<wstring::const\\_iterator>](#) **std::wsmatch**
- typedef [regex\\_iterator<wstring::const\\_iterator>](#) **std::wsregex\_iterator**
- typedef [regex\\_token\\_iterator<wstring::const\\_iterator>](#) **std::wsregex\_token\_iterator**
- typedef [sub\\_match<wstring::const\\_iterator>](#) **std::wssub\_match**

## Enumerations

- enum class **\_RegexExecutorPolicy** : int { **\_S\_auto** , **\_S\_alternate** }

## Functions

- template<typename \_Bilter, typename \_Alloc, typename \_CharT, typename \_TraitsT>  
bool **std::detail::regex\_algo\_impl** (\_Bilter \_\_s, \_Bilter \_\_e, [match\\_results](#)< \_Bilter, \_Alloc > &\_\_m, const [basic\\_regex](#)< \_CharT, \_TraitsT > &\_\_re, [regex\\_constants::match\\_flag\\_type](#) \_\_flags, \_RegexExecutorPolicy \_\_policy, bool \_\_match\_mode)
- template<typename \_ForwardIterator>  
**std::basic\_regex** (\_ForwardIterator, \_ForwardIterator, [regex\\_constants::syntax\\_option\\_type](#)={}) -> [basic\\_regex](#)< typename [iterator\\_traits](#)< \_ForwardIterator >::value\_type >

## Matching, Searching, and Replacing

- template<typename \_Bi\_iter, typename \_Ch\_type, typename \_Rx\_traits>  
bool **std::regex\_match** (\_Bi\_iter \_\_first, \_Bi\_iter \_\_last, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_re, [regex\\_constants::match\\_flag\\_type](#) \_\_flags=[regex\\_constants::match\\_default](#))
- template<typename \_Bi\_iter, typename \_Alloc, typename \_Ch\_type, typename \_Rx\_traits>  
bool **std::regex\_match** (\_Bi\_iter \_\_s, \_Bi\_iter \_\_e, [match\\_results](#)< \_Bi\_iter, \_Alloc > &\_\_m, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_re, [regex\\_constants::match\\_flag\\_type](#) \_\_flags=[regex\\_constants::match\\_default](#))
- template<typename \_Ch\_type, class \_Rx\_traits>  
bool **std::regex\_match** (const \_Ch\_type \*\_\_s, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_re, [regex\\_constants::match\\_flag\\_type](#) \_\_f=[regex\\_constants::match\\_default](#))
- template<typename \_Ch\_type, typename \_Alloc, typename \_Rx\_traits>  
bool **std::regex\_match** (const \_Ch\_type \*\_\_s, [match\\_results](#)< const \_Ch\_type \*, \_Alloc > &\_\_m, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_re, [regex\\_constants::match\\_flag\\_type](#) \_\_f=[regex\\_constants::match\\_default](#))
- template<typename \_Ch\_traits, typename \_Ch\_alloc, typename \_Alloc, typename \_Ch\_type, typename \_Rx\_traits>  
bool **std::regex\_match** (const [basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc > &&, [match\\_results](#)< typename [basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc >::const\_iterator, \_Alloc > &, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))=delete
- template<typename \_Ch\_traits, typename \_Ch\_alloc, typename \_Alloc, typename \_Ch\_type, typename \_Rx\_traits>  
bool **std::regex\_match** (const [basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc > &\_\_s, [match\\_results](#)< typename [basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc >::const\_iterator, \_Alloc > &\_\_m, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_re, [regex\\_constants::match\\_flag\\_type](#) \_\_flags=[regex\\_constants::match\\_default](#))
- template<typename \_Ch\_traits, typename \_Str\_allocator, typename \_Ch\_type, typename \_Rx\_traits>  
bool **std::regex\_match** (const [basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Str\_allocator > &\_\_s, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_re, [regex\\_constants::match\\_flag\\_type](#) \_\_flags=[regex\\_constants::match\\_default](#))
- template<typename \_Out\_iter, typename \_Bi\_iter, typename \_Rx\_traits, typename \_Ch\_type>  
\_Out\_iter **std::regex\_replace** (\_Out\_iter \_\_out, \_Bi\_iter \_\_first, \_Bi\_iter \_\_last, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_e, const \_Ch\_type \*\_\_fmt, [regex\\_constants::match\\_flag\\_type](#) \_\_flags=[regex\\_constants::match\\_default](#))
- template<typename \_Out\_iter, typename \_Bi\_iter, typename \_Rx\_traits, typename \_Ch\_type, typename \_St, typename \_Sa>  
\_Out\_iter **std::regex\_replace** (\_Out\_iter \_\_out, \_Bi\_iter \_\_first, \_Bi\_iter \_\_last, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_e, const [basic\\_string](#)< \_Ch\_type, \_St, \_Sa > &\_\_fmt, [regex\\_constants::match\\_flag\\_type](#) \_\_flags=[regex\\_constants::match\\_default](#))
- template<typename \_Rx\_traits, typename \_Ch\_type>  
[basic\\_string](#)< \_Ch\_type > **std::regex\_replace** (const \_Ch\_type \*\_\_s, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_e, const \_Ch\_type \*\_\_fmt, [regex\\_constants::match\\_flag\\_type](#) \_\_flags=[regex\\_constants::match\\_default](#))
- template<typename \_Rx\_traits, typename \_Ch\_type, typename \_St, typename \_Sa>  
[basic\\_string](#)< \_Ch\_type > **std::regex\_replace** (const \_Ch\_type \*\_\_s, const [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_e, const [basic\\_string](#)< \_Ch\_type, \_St, \_Sa > &\_\_fmt, [regex\\_constants::match\\_flag\\_type](#) \_\_flags=[regex\\_constants::match\\_default](#))
- template<typename \_Rx\_traits, typename \_Ch\_type, typename \_St, typename \_Sa>  
[basic\\_string](#)< \_Ch\_type, \_St, \_Sa > **std::regex\_replace** (const [basic\\_string](#)< \_Ch\_type, \_St, \_Sa > &\_\_s,

```

const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type
__flags=regex_constants::match_default)
• template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>
basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s,
const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,
regex_constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re,
regex_constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const
basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Ch_type, typename _Rx_traits>
bool std::regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > &__e,
regex_constants::match_flag_type __f=regex_constants::match_default)
• template<typename _Ch_type, class _Alloc, class _Rx_traits>
bool std::regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > &__m,
const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=
regex_constants::match_default)
• template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< type-
name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type,
_Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
• template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< type-
name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex<
_Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
• template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const
basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)

```

### 6.122.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.123 regex.tcc File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Functions

- template<typename \_Out\_iter, typename \_Bi\_iter, typename \_Rx\_traits, typename \_Ch\_type>
\_Out\_iter **std::regex\_replace** (\_Out\_iter \_\_out, \_Bi\_iter \_\_first, \_Bi\_iter \_\_last, const basic\_regex< \_Ch\_type,
\_Rx\_traits > &\_\_e, const \_Ch\_type \* \_\_fmt, size\_t \_\_len, regex\_constants::match\_flag\_type \_\_flags)

### 6.123.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.124 `regex_automaton.h` File Reference

### Classes

- class `std::__detail::_StateSeq<_TraitsT>`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Macros

- `#define _GLIBCXX_REGEX_STATE_LIMIT`

### Typedefs

- `template<typename _CharT>`  
using `std::__detail::_Matcher`
- `typedef long std::__detail::_StatIdT`

### Enumerations

- enum `std::__detail::_Opcode` : int {  
    `_S_opcode_unknown` , `_S_opcode_alternative` , `_S_opcode_repeat` , `_S_opcode_backref` ,  
    `_S_opcode_line_begin_assertion` , `_S_opcode_line_end_assertion` , `_S_opcode_word_boundary` , `_S_opcode_subexpr_lookahead` ,  
    `_S_opcode_subexpr_begin` , `_S_opcode_subexpr_end` , `_S_opcode_dummy` , `_S_opcode_match` ,  
    `_S_opcode_accept` }

### Variables

- `constexpr _StatIdT std::__detail::_S_invalid_state_id`

#### 6.124.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.125 `regex_automaton.tcc` File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

#### 6.125.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.126 `regex_compiler.h` File Reference

### Classes

- struct `std::__detail::_BracketMatcher<_TraitsT, __icase, __collate>`
- class `std::__detail::_Compiler<_TraitsT>`

## Namespaces

- namespace `std`
- namespace `std::__detail`

### 6.126.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.127 `regex_compiler.tcc` File Reference

## Namespaces

- namespace `std`
- namespace `std::__detail`

## Macros

- `#define __INSERT_REGEX_MATCHER(__func, ...)`

### 6.127.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.128 `regex_constants.h` File Reference

## Namespaces

- namespace `std`
- namespace `std::regex_constants`

## 5.1 Regular Expression Syntax Options

- constexpr `syntax_option_type std::regex_constants::__multiline`
- constexpr `syntax_option_type std::regex_constants::__polynomial`
- constexpr `syntax_option_type std::regex_constants::awk`
- constexpr `syntax_option_type std::regex_constants::basic`
- constexpr `syntax_option_type std::regex_constants::collate`
- constexpr `syntax_option_type std::regex_constants::ECMAScript`
- constexpr `syntax_option_type std::regex_constants::egrep`
- constexpr `syntax_option_type std::regex_constants::extended`
- constexpr `syntax_option_type std::regex_constants::grep`
- constexpr `syntax_option_type std::regex_constants::icase`
- constexpr `syntax_option_type std::regex_constants::multiline`
- constexpr `syntax_option_type std::regex_constants::nosubs`
- constexpr `syntax_option_type std::regex_constants::operator& (syntax_option_type __a, syntax_option_type __b) noexcept`
- constexpr `syntax_option_type & std::regex_constants::operator&= (syntax_option_type &__a, syntax_option_type __b) noexcept`
- constexpr `syntax_option_type std::regex_constants::operator^ (syntax_option_type __a, syntax_option_type __b) noexcept`
- constexpr `syntax_option_type & std::regex_constants::operator^= (syntax_option_type &__a, syntax_option_type __b) noexcept`



- constexpr `syntax_option_type std::regex_constants::operator|` (`syntax_option_type __a`, `syntax_option_type __b`) noexcept
- constexpr `syntax_option_type & std::regex_constants::operator|=` (`syntax_option_type &__a`, `syntax_option_type __b`) noexcept
- constexpr `syntax_option_type std::regex_constants::operator~` (`syntax_option_type __a`) noexcept
- constexpr `syntax_option_type std::regex_constants::optimize`
- enum `std::regex_constants::syntax_option_type` : unsigned int {  
`_S_icode` , `_S_nosubs` , `_S_optimize` , `_S_collate` ,  
`_S_ECMAScript` , `_S_basic` , `_S_extended` , `_S_awk` ,  
`_S_grep` , `_S_egrep` , `_S_polynomial` , `_S_multiline` }

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- constexpr `match_flag_type std::regex_constants::format_default`
- constexpr `match_flag_type std::regex_constants::format_first_only`
- constexpr `match_flag_type std::regex_constants::format_no_copy`
- constexpr `match_flag_type std::regex_constants::format_sed`
- constexpr `match_flag_type std::regex_constants::match_any`
- constexpr `match_flag_type std::regex_constants::match_continuous`
- constexpr `match_flag_type std::regex_constants::match_default`
- enum `std::regex_constants::match_flag_type` : unsigned int {  
`_S_default` , `_S_not_bol` , `_S_not_eol` , `_S_not_bow` ,  
`_S_not_eow` , `_S_any` , `_S_not_null` , `_S_continuous` ,  
`_S_prev_avail` , `_S_sed` , `_S_no_copy` , `_S_first_only` ,  
`_S_match_flag_last` }
- constexpr `match_flag_type std::regex_constants::match_not_bol`
- constexpr `match_flag_type std::regex_constants::match_not_bow`
- constexpr `match_flag_type std::regex_constants::match_not_eol`
- constexpr `match_flag_type std::regex_constants::match_not_eow`
- constexpr `match_flag_type std::regex_constants::match_not_null`
- constexpr `match_flag_type std::regex_constants::match_prev_avail`
- constexpr `match_flag_type std::regex_constants::operator&` (`match_flag_type __a`, `match_flag_type __b`) noexcept
- constexpr `match_flag_type & std::regex_constants::operator&=` (`match_flag_type &__a`, `match_flag_type __b`) noexcept
- constexpr `match_flag_type std::regex_constants::operator^` (`match_flag_type __a`, `match_flag_type __b`) noexcept
- constexpr `match_flag_type & std::regex_constants::operator^=` (`match_flag_type &__a`, `match_flag_type __b`) noexcept
- constexpr `match_flag_type std::regex_constants::operator|` (`match_flag_type __a`, `match_flag_type __b`) noexcept
- constexpr `match_flag_type & std::regex_constants::operator|=` (`match_flag_type &__a`, `match_flag_type __b`) noexcept
- constexpr `match_flag_type std::regex_constants::operator~` (`match_flag_type __a`) noexcept

### 6.128.1 Detailed Description

Constant definitions for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.129 `regex_error.h` File Reference

### Classes

- class `std::regex_error`

### Namespaces

- namespace `std`
- namespace `std::regex_constants`

### 5.3 Error Types

- constexpr `error_type` `std::regex_constants::error_backref` (`_S_error_backref`)
- constexpr `error_type` `std::regex_constants::error_badbrace` (`_S_error_badbrace`)
- constexpr `error_type` `std::regex_constants::error_badrepeat` (`_S_error_badrepeat`)
- constexpr `error_type` `std::regex_constants::error_brace` (`_S_error_brace`)
- constexpr `error_type` `std::regex_constants::error_brack` (`_S_error_brack`)
- constexpr `error_type` `std::regex_constants::error_collate` (`_S_error_collate`)
- constexpr `error_type` `std::regex_constants::error_complexity` (`_S_error_complexity`)
- constexpr `error_type` `std::regex_constants::error_ctype` (`_S_error_ctype`)
- constexpr `error_type` `std::regex_constants::error_escape` (`_S_error_escape`)
- constexpr `error_type` `std::regex_constants::error_paren` (`_S_error_paren`)
- constexpr `error_type` `std::regex_constants::error_range` (`_S_error_range`)
- constexpr `error_type` `std::regex_constants::error_space` (`_S_error_space`)
- constexpr `error_type` `std::regex_constants::error_stack` (`_S_error_stack`)
- enum `std::regex_constants::error_type` {  
`_S_error_collate` , `_S_error_ctype` , `_S_error_escape` , `_S_error_backref` ,  
`_S_error_brack` , `_S_error_paren` , `_S_error_brace` , `_S_error_badbrace` ,  
`_S_error_range` , `_S_error_space` , `_S_error_badrepeat` , `_S_error_complexity` ,  
`_S_error_stack` , `_S_null` , `_S_grammar` }

#### 6.129.1 Detailed Description

Error and exception objects for the `std` regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.130 `regex_executor.h` File Reference

### Classes

- class `std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode >`

### Namespaces

- namespace `std`
- namespace `std::__detail`

#### 6.130.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.131 `regex_executor.tcc` File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

#### 6.131.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.132 `regex_scanner.h` File Reference

### Classes

- class `std::__detail::_Scanner<_CharT>`

### Namespaces

- namespace `std`
- namespace `std::__detail`

#### 6.132.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.133 `regex_scanner.tcc` File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

#### 6.133.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.134 `requires_hosted.h` File Reference

#### 6.134.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<version>`.

## 6.135 `sat_arith.h` File Reference

#### 6.135.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

## 6.136 semaphore\_base.h File Reference

### 6.136.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<semaphore>`.

## 6.137 bits/shared\_ptr.h File Reference

### Classes

- class [std::enable\\_shared\\_from\\_this<\\_Tp>](#)
- struct [std::hash< shared\\_ptr<\\_Tp>>](#)
- struct [std::owner\\_less< shared\\_ptr<\\_Tp>>](#)
- struct [std::owner\\_less< void>](#)
- struct [std::owner\\_less< weak\\_ptr<\\_Tp>>](#)
- class [std::shared\\_ptr<\\_Tp>](#)
- class [std::weak\\_ptr<\\_Tp>](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- template<typename \_Del, typename \_Tp, \_Lock\_policy \_Lp>  
\_Del \* [std::get\\_deleter](#) (const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_p) noexcept
- template<typename \_Tp, typename \_Del>  
[std::shared\\_ptr](#) ([unique\\_ptr](#)<\_Tp, \_Del>) -> shared\_ptr<\_Tp>
- template<typename \_Tp>  
[std::shared\\_ptr](#) ([weak\\_ptr](#)<\_Tp>) -> shared\_ptr<\_Tp>
- template<typename \_Tp>  
[std::weak\\_ptr](#) ([shared\\_ptr](#)<\_Tp>) -> weak\_ptr<\_Tp>

### 6.137.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.138 experimental/bits/shared\_ptr.h File Reference

### Classes

- struct [std::hash< experimental::shared\\_ptr<\\_Tp>>](#)
- struct [std::experimental::fundamentals\\_v2::owner\\_less< shared\\_ptr<\\_Tp>>](#)
- struct [std::experimental::fundamentals\\_v2::owner\\_less< weak\\_ptr<\\_Tp>>](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

## Functions

- `template<typename _Tp>`  
`bool std::experimental::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > * __v, shared_ptr< _Tp > __w)`
- `template<typename _Tp>`  
`bool std::experimental::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > * __v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp>`  
`bool std::experimental::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > * __v, shared_ptr< _Tp > __w)`
- `template<typename _Tp>`  
`bool std::experimental::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > * __v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp>`  
`void std::experimental::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::experimental::atomic_exchange_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)`
- `template<typename _Tp>`  
`bool std::experimental::atomic_is_lock_free (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::experimental::atomic_load (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::experimental::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order __mo)`
- `template<typename _Tp>`  
`void std::experimental::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::experimental::atomic_store_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > std::experimental::const_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > std::experimental::dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Del, typename _Tp>`  
`_Del * std::experimental::get_deleter (const shared_ptr< _Tp > &__p) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Ch, typename _Tr, typename _Tp>`  
`std::basic_ostream< _Ch, _Tr > & std::experimental::operator<< (std::basic_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)`

- `template<typename _Tp>`  
`bool std::experimental::operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > std::experimental::reinterpret_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > std::experimental::static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp>`  
`void std::experimental::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp>`  
`void std::experimental::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

## Variables

- `template<typename _Yp, typename _Tp>`  
`constexpr bool std::experimental::__sp_compatible_v`
- `template<typename _Tp, typename _Yp>`  
`constexpr bool std::experimental::__sp_is_constructible_v`

### 6.138.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/memory>`.

## 6.139 shared\_ptr\_atomic.h File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_TSAN_MUTEX_DESTROY(X)`
- `#define _GLIBCXX_TSAN_MUTEX_LOCKED(X)`
- `#define _GLIBCXX_TSAN_MUTEX_POST_SIGNAL(X)`
- `#define _GLIBCXX_TSAN_MUTEX_POST_UNLOCK(X)`
- `#define _GLIBCXX_TSAN_MUTEX_PRE_SIGNAL(X)`
- `#define _GLIBCXX_TSAN_MUTEX_PRE_UNLOCK(X)`
- `#define _GLIBCXX_TSAN_MUTEX_TRY_LOCK(X)`
- `#define _GLIBCXX_TSAN_MUTEX_TRY_LOCK_FAILED(X)`

## Functions

- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic\_is\_lock\_free (const __shared_ptr< _Tp, _Lp > *)`
- `template<typename _Tp>`  
`bool std::atomic\_is\_lock\_free (const shared\_ptr< _Tp > *__p)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic\_load (const __shared_ptr< _Tp, _Lp > *__p)`
- `template<typename _Tp>`  
`shared\_ptr< _Tp > std::atomic\_load (const shared\_ptr< _Tp > *__p)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic\_load\_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory\_order)`
- `template<typename _Tp>`  
`shared\_ptr< _Tp > std::atomic\_load\_explicit (const shared\_ptr< _Tp > *__p, memory\_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic\_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp>`  
`void std::atomic\_store (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic\_store\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory\_order)`
- `template<typename _Tp>`  
`void std::atomic\_store\_explicit (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > __r, memory\_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic\_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp>`  
`shared\_ptr< _Tp > std::atomic\_exchange (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic\_exchange\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory\_order)`
- `template<typename _Tp>`  
`shared\_ptr< _Tp > std::atomic\_exchange\_explicit (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > __r, memory\_order)`

- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`

### 6.139.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.140 shared\_ptr\_base.h File Reference

### Classes

- struct `std::_Sp_ebo_helper< _Nm, _Tp, false >`
- struct `std::_Sp_ebo_helper< _Nm, _Tp, true >`
- class `std::bad_weak_ptr`
- struct `std::hash< __shared_ptr< _Tp, _Lp > >`

### Namespaces

- namespace `std`

### Enumerations

- enum `_Lock_policy`

### Functions

- `template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename _Alloc, typename... _Args>`  
`__shared_ptr< _Tp, _Lp > std::__allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename... _Args>`  
`__shared_ptr< _Tp, _Lp > std::make_shared (_Args &&... __args)`



- `template<typename _Tp>`  
`_Tp * std::__shared_ptr_deref (_Tp *__p)`
- `void std::__throw_bad_weak_ptr ()`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename _Up, _Lock_policy _Lp>`  
`strong_ordering std::operator<=> (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`strong_ordering std::operator<=> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool std::operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::reinterpret_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`

## Variables

- `const _Lock_policy std::__default_lock_policy`

### 6.140.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.141 `slice_array.h` File Reference

### Classes

- class `std::slice`
- class `std::slice_array< _Tp >`

### Namespaces

- namespace `std`

### 6.141.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.142 specfun.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)`
- `float __gnu_cxx::airy_aif (float __x)`
- `long double __gnu_cxx::airy_ail (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)`
- `float __gnu_cxx::airy_bif (float __x)`
- `long double __gnu_cxx::airy_bil (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tpa, typename _Tpb>`  
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)`
- `float std::betaf (float __a, float __b)`
- `long double std::betal (long double __a, long double __b)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)`
- `float std::comp_ellint_1f (float __k)`
- `long double std::comp_ellint_1l (long double __k)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)`
- `float std::comp_ellint_2f (float __k)`
- `long double std::comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn>`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (_Tp __k, _Tpn __nu)`
- `float std::comp_ellint_3f (float __k, float __nu)`
- `long double std::comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)`
- `long double __gnu_cxx::conf_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (_Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_if (float __nu, float __x)`
- `long double std::cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (_Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_jf (float __nu, float __x)`
- `long double std::cyl_bessel_jl (long double __nu, long double __x)`

- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_kf (float __nu, float __x)`
- `long double std::cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_neumannf (float __nu, float __x)`
- `long double std::cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_1f (float __k, float __phi)`
- `long double std::ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_2f (float __k, float __phi)`
- `long double std::ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp>`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi)`
- `float std::ellint_3f (float __k, float __nu, float __phi)`
- `long double std::ellint_3l (long double __k, long double __nu, long double __phi)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::expint ( _Tp __x)`
- `float std::expintf (float __x)`
- `long double std::expintl (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::hermite (unsigned int __n, _Tp __x)`
- `float std::hermitef (unsigned int __n, float __x)`
- `long double std::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg ( _Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::hypergf (float __a, float __b, float __c, float __x)`
- `long double __gnu_cxx::hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::laguerre (unsigned int __n, _Tp __x)`
- `float std::laguerref (unsigned int __n, float __x)`
- `long double std::laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::legendre (unsigned int __l, _Tp __x)`
- `float std::legendref (unsigned int __l, float __x)`
- `long double std::legendrel (unsigned int __l, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::riemann_zeta ( _Tp __s)`
- `float std::riemann_zetaf (float __s)`
- `long double std::riemann_zetal (long double __s)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::sph_bessel (unsigned int __n, _Tp __x)`
- `float std::sph_besself (unsigned int __n, float __x)`
- `long double std::sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float std::sph_legendref (unsigned int __l, unsigned int __m, float __theta)`

- long double [std::sph\\_legendrel](#) (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_neumann](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::sph\\_neumannf](#) (unsigned int \_\_n, float \_\_x)
- long double [std::sph\\_neumannl](#) (unsigned int \_\_n, long double \_\_x)

### 6.142.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cmath>`.

## 6.143 sstream.tcc File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _SSTREAM_TCC`

### 6.143.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<sstream>`.

## 6.144 std\_abs.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- constexpr double [std::abs](#) (double \_\_x)
- constexpr float [std::abs](#) (float \_\_x)
- long [std::abs](#) (long \_\_i)
- constexpr long double [std::abs](#) (long double \_\_x)
- long long [std::abs](#) (long long \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::abs(declval<double>())), \_Tp, \_Abi>>  
\_\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > [std::abs](#) (simd< \_Tp, \_Abi > \_\_x)

### 6.144.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cmath>` or `<cstdlib>`.

## 6.145 std\_function.h File Reference

### Classes

- struct [std::\\_\\_is\\_location\\_invariant< \\_Tp >](#)
- class [std::\\_Function\\_base](#)
- class [std::bad\\_function\\_call](#)
- class [std::function< \\_Res\(\\_ArgTypes...\)>](#)

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

## Typedefs

- `template<typename _Fn, typename _Op>`  
`using std::\_\_function\_guide\_t`

## Enumerations

- `enum \_Manager\_operation { \_\_get\_type\_info , \_\_get\_func\_ptr , \_\_clone\_func , \_\_destroy\_func }`

## Functions

- `template<typename _Fn, typename _Signature = \_\_function\_guide\_t<_Fn, decltype(&_Fn::operator())>>`  
`std::function (_Fn) -> function<_Signature >`
- `template<typename _Res, typename... _ArgTypes>`  
`std::function (_Res(*)(_ArgTypes...)) -> function<_Res(_ArgTypes...)>`
- `template<typename _Res, typename... _Args>`  
`bool std::operator== (const function<_Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _Res, typename... _Args>`  
`void std::swap (function<_Res(_Args...)> &__x, function<_Res(_Args...)> &__y) noexcept`

### 6.145.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.146 [std\\_mutex.h](#) File Reference

### Classes

- struct [std::adopt\\_lock\\_t](#)
- struct [std::defer\\_lock\\_t](#)
- class [std::lock\\_guard](#)< [\\_Mutex](#) >
- class [std::mutex](#)
- struct [std::try\\_to\\_lock\\_t](#)

### Namespaces

- namespace [std](#)

### Variables

- constexpr [adopt\\_lock\\_t](#) [std::adopt\\_lock](#)
- constexpr [defer\\_lock\\_t](#) [std::defer\\_lock](#)
- constexpr [try\\_to\\_lock\\_t](#) [std::try\\_to\\_lock](#)

### 6.146.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<mutex>`.

## 6.147 std\_thread.h File Reference

### Classes

- struct [std::hash< thread::id >](#)
- class [std::thread::id](#)
- class [std::thread](#)

### Namespaces

- namespace [std](#)
- namespace [std::this\\_thread](#)

### Functions

- [thread::id std::this\\_thread::get\\_id \(\)](#) noexcept
- void [std::this\\_thread::yield \(\)](#) noexcept

#### 6.147.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<thread>`.

## 6.148 stl\_algo.h File Reference

### Namespaces

- namespace [std](#)

### Enumerations

- enum { [\\_S\\_chunk\\_size](#) }

### Functions

- template<typename [\\_ForwardIterator](#), typename [\\_BinaryPredicate](#)>  
constexpr [\\_ForwardIterator](#) **std::adjacent\_find** ([\\_ForwardIterator](#) \_\_first, [\\_ForwardIterator](#) \_\_last, [\\_BinaryPredicate](#) \_\_binary\_pred)
- template<typename [\\_RandomAccessIterator](#), typename [\\_Distance](#), typename [\\_Compare](#)>  
constexpr void **std::chunk\_insertion\_sort** ([\\_RandomAccessIterator](#) \_\_first, [\\_RandomAccessIterator](#) \_\_last, [\\_Distance](#) \_\_chunk\_size, [\\_Compare](#) \_\_comp)
- template<typename [\\_ForwardIterator](#), typename [\\_Tp](#), typename [\\_CompareItTp](#), typename [\\_CompareTpIt](#)>  
constexpr [pair](#)<[\\_ForwardIterator](#), [\\_ForwardIterator](#)> **std::equal\_range** ([\\_ForwardIterator](#) \_\_first, [\\_ForwardIterator](#) \_\_last, const [\\_Tp](#) &\_\_val, [\\_CompareItTp](#) \_\_comp\_it\_val, [\\_CompareTpIt](#) \_\_comp\_val\_it)
- template<typename [\\_BidirectionalIterator1](#), typename [\\_BidirectionalIterator2](#), typename [\\_BinaryPredicate](#)>  
constexpr [\\_BidirectionalIterator1](#) **std::find\_end** ([\\_BidirectionalIterator1](#) \_\_first1, [\\_BidirectionalIterator1](#) \_\_last1, [\\_BidirectionalIterator2](#) \_\_first2, [\\_BidirectionalIterator2](#) \_\_last2, [bidirectional\\_iterator\\_tag](#), [bidirectional\\_iterator\\_tag](#), [\\_BinaryPredicate](#) \_\_comp)
- template<typename [\\_ForwardIterator1](#), typename [\\_ForwardIterator2](#), typename [\\_BinaryPredicate](#)>  
constexpr [\\_ForwardIterator1](#) **std::find\_end** ([\\_ForwardIterator1](#) \_\_first1, [\\_ForwardIterator1](#) \_\_last1, [\\_ForwardIterator2](#) \_\_first2, [\\_ForwardIterator2](#) \_\_last2, [forward\\_iterator\\_tag](#), [forward\\_iterator\\_tag](#), [\\_BinaryPredicate](#) \_\_comp)
- template<typename [\\_InputIterator](#), typename [\\_Predicate](#)>  
constexpr [\\_InputIterator](#) **std::find\_if\_not** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [\\_Predicate](#) \_\_pred)
- template<typename [\\_InputIterator](#), typename [\\_Predicate](#), typename [\\_Distance](#)>  
constexpr [\\_InputIterator](#) **std::find\_if\_not\_n** ([\\_InputIterator](#) \_\_first, [\\_Distance](#) &\_\_len, [\\_Predicate](#) \_\_pred)

- `template<typename _EuclideanRingElement>`  
`constexpr _EuclideanRingElement std::gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _IntType, typename _UniformRandomBitGenerator>`  
`pair< _IntType, _IntType > std::gen_two_uniform_ints (_IntType __b0, _IntType __b1, _UniformRandomBitGenerator &&__g)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare>`  
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`void std::inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>`  
`void std::merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>`  
`void std::merge_adaptive_resize (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare>`  
`void std::merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>`  
`void std::merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare>`  
`void std::merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iterator, typename _Compare>`  
`constexpr void std::move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare>`  
`_OutputIterator std::move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2, _OutputIterator __result, _Compare __comp)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare>`  
`void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, ↵`  
`_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::__next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare`  
`__comp)`
- `template<typename _BidirectionalIterator, typename _Predicate>`  
`constexpr _BidirectionalIterator std::__partition (_BidirectionalIterator __first, _BidirectionalIterator __last, ↵`  
`_Predicate __pred, bidirectional\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred,`  
`forward\_iterator\_tag)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::__prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare`  
`__comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator std::__remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator`  
`__result, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>`  
`constexpr _OutputIterator std::__replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator`  
`__result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator>`  
`constexpr void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator>`  
`constexpr _BidirectionalIterator std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, ↵`  
`_BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator`  
`__last, forward\_iterator\_tag)`
- `template<typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __↵`  
`middle, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance>`  
`_BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, ↵`  
`_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance`  
`__buffer_size)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBit↵`  
`Generator>`  
`_OutputIterator std::__sample (_ForwardIterator __first, _ForwardIterator __last, forward\_iterator\_tag, _Output↵`  
`Iterator __out, _Cat, _Size __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator>`  
`_RandomAccessIterator std::__sample (_InputIterator __first, _InputIterator __last, input\_iterator\_tag, ↵`  
`_RandomAccessIterator __out, random\_access\_iterator\_tag, _Size __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count,`  
`_UnaryPredicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator std::__search_n_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer ↵`  
`__count, _UnaryPredicate __unary_pred, std::forward\_iterator\_tag)`



- `template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate>`  
`constexpr _RandomAccessIter std::\_\_search\_n\_aux (_RandomAccessIter __first, _RandomAccessIter __last, ↵`  
`_Integer __count, _UnaryPredicate __unary_pred, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::\_\_set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`  
`__first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::\_\_set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, ↵`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::\_\_set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, ↵`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::\_\_set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 ↵`  
`__first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`_ForwardIterator std::\_\_stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>`  
`_ForwardIterator std::\_\_stable\_partition\_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate`  
`__pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`void std::\_\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>`  
`void std::\_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __middle, ↵`  
`_RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare>`  
`void std::\_\_stable\_sort\_adaptive\_resize (_RandomAccessIterator __first, _RandomAccessIterator __last, ↵`  
`_Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::\_\_unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator std::\_\_unique\_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator`  
`__result, _BinaryPredicate __binary_pred, forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator ↵`  
`__result, _BinaryPredicate __binary_pred, input\_iterator\_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>`  
`_ForwardIterator std::\_\_unique\_copy\_1 (_InputIterator __first, _InputIterator __last, _ForwardIterator __result,`  
`_BinaryPredicate __binary_pred, __true_type)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator std::\_\_unique\_copy\_1 (_InputIterator __first, _InputIterator __last, _OutputIterator ↵`  
`__result, _BinaryPredicate __binary_pred, __false_type)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::\_\_upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp`  
`& __val, _Compare __comp)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator std::copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, \_P Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`constexpr _OutputIterator std::copy\_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::count\_if (_InputIterator __first, _InputIterator \_P __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator __first, \_ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator __first, \_ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, \_ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, \_ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, \_ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, \_ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function>`  
`constexpr _Function std::for\_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _InputIterator, typename _Size, typename _Function>`  
`constexpr _InputIterator std::for\_each\_n (_InputIterator __first, _Size __n, _Function __f)`
- `template<typename _ForwardIterator, typename _Generator>`  
`constexpr void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`

- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`constexpr _OutputIterator std::generate\_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare>`  
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator>`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::is\_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator>`  
`constexpr bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp std::max (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp std::max (initializer\_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

- `template<typename _Tp>`  
`constexpr _Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator>`  
`constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>`  
`constexpr pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`

- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::partition\_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _BidirectionalIterator>`  
`constexpr bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator std::remove\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator std::replace\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>`  
`constexpr _OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp>`  
`constexpr void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator>`  
`constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomBitGenerator>`  
`_SampleIterator std::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`constexpr _ForwardIterator std::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`

- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp>`  
`constexpr _ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator>`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>`  
`constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`

- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵  
binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,  
_BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↵  
val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↵  
val, _Compare __comp)`

### 6.148.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 6.149 `std_algobase.h` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define \_GLIBCXX\_ADVANCE(P, N)`
- `#define \_GLIBCXX\_MOVE3(_Tp, _Up, _Vp)`
- `#define \_GLIBCXX\_MOVE\_BACKWARD3(_Tp, _Up, _Vp)`
- `#define \_GLIBCXX\_TO\_ADDR(P)`

### Functions

- `template<bool _IsMove, typename _OutIter, typename _InIter>`  
`constexpr void std::\_\_assign\_one (_OutIter &__out, _InIter &__in)`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI std::\_\_copy\_move\_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr \_\_gnu\_debug::\_\_Safe\_iterator< _Ite, _Seq, _Cat > std::\_\_copy\_move\_a (_II, _II, const  
::\_\_gnu\_debug::\_\_Safe\_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>`  
`constexpr ::\_\_gnu\_debug::\_\_Safe\_iterator< _OIte, _OSeq, _OCat > std::\_\_copy\_move\_a (const ::\_\_gnu\_debug::\_\_Safe\_iterator<  
_IIte, _ISeq, _ICat > &, const ::\_\_gnu\_debug::\_\_Safe\_iterator< _IIte, _ISeq, _ICat > &, const ::\_\_gnu\_debug::\_\_Safe\_iterator<  
_OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI std::\_\_copy\_move\_a (const ::\_\_gnu\_debug::\_\_Safe\_iterator< _Ite, _Seq, _Cat > &, const  
::\_\_gnu\_debug::\_\_Safe\_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>`  
`::Deque\_iterator< _OTp, _OTp &, _OTp * > std::\_\_copy\_move\_a1 (::Deque_iterator< _ITp, _IRef, _IPtr >  
__first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result)`



- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>`  
`_OI std::__copy_move_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr >`  
`__last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Tp>`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, ::Deque_iterator< _Tp, _Tp &, _Tp * >`  
`>::__type std::__copy_move_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI std::__copy_move_a1 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT >`  
`> >::__type std::__copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT`  
`> >)`
- `template<bool _IsMove, typename _InIter, typename _Sent, typename _OutIter>`  
`constexpr _OutIter std::__copy_move_a2 (_InIter __first, _Sent __last, _OutIter __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT`  
`> >::__type std::__copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT,`  
`char_traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ::Deque_iterator< _CharT, _CharT &, _CharT`  
`* > >::__type std::__copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT > > __first,`  
`istreambuf_iterator< _CharT, char_traits< _CharT > > __last, ::Deque_iterator< _CharT, _CharT &, _CharT`  
`* > __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2`  
`(istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT`  
`> >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI std::__copy_move_backward_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr __gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > std::__copy_move_backward_a (_II, _II, const`  
`::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _Ite, typename _ISeq, typename _ICat, typename _Olte, typename _OSeq, typename _OCat>`  
`constexpr ::__gnu_debug::Safe_iterator< _Olte, _OSeq, _OCat > std::__copy_move_backward_a (const`  
`::__gnu_debug::Safe_iterator< _Ite, _ISeq, _ICat > &, const ::__gnu_debug::Safe_iterator< _Ite, _ISeq, _ICat`  
`> &, const ::__gnu_debug::Safe_iterator< _Olte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI std::__copy_move_backward_a (const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &,`  
`const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > std::__copy_move_backward_a1 (::Deque_iterator< _ITp, _IRef,`  
`_IPtr > __first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp *`  
`> __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>`  
`_OI std::__copy_move_backward_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp,`  
`_Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2>`  
`constexpr _BI2 std::__copy_move_backward_a1 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _II, typename _Tp>`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, ::Deque_iterator< _Tp, _Tp &, _Tp * >`  
`>::__type std::__copy_move_backward_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * >`  
`__result)`



- `template<bool _IsMove, typename _BI1, typename _BI2>`  
`constexpr _BI2 std::__copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`constexpr _OutputIterator std::__copy_n_a (_InputIterator __first, _Size __n, _OutputIterator __result, bool)`
- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, __Deque_iterator< _CharT, _CharT &, _CharT * >`  
`>::__type std::__copy_n_a (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size, __`  
`Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)`
- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_n_a (istreambuf_iterator<`  
`_CharT, char_traits< _CharT > >, _Size, _CharT *, bool)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr iterator_traits< _InputIterator >::difference_type std::__count_if (_InputIterator __first, _InputIterator`  
`__last, _Predicate __pred)`
- `template<typename _II1, typename _II2>`  
`constexpr bool std::__equal4 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _BinaryPredicate>`  
`constexpr bool std::__equal4 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _BinaryPredicate __binary,`  
`_pred)`
- `template<typename _II1, typename _II2>`  
`constexpr bool std::__equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _II2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::__equal_aux (_II1, _II1, const __gnu_debug::Safe_iterator< _II2, _Seq2, _Cat2 > &)`
- `template<typename _II1, typename _Seq1, typename _Cat1, typename _II2>`  
`constexpr bool std::__equal_aux (const __gnu_debug::Safe_iterator< _II1, _Seq1, _Cat1 > &, const`  
`__gnu_debug::Safe_iterator< _II1, _Seq1, _Cat1 > &, _II2)`
- `template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::__equal_aux (const __gnu_debug::Safe_iterator< _II1, _Seq1, _Cat1 > &, const`  
`__gnu_debug::Safe_iterator< _II1, _Seq1, _Cat1 > &, const __gnu_debug::Safe_iterator< _II2, _Seq2,`  
`_Cat2 > &)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II>`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type std::__equal_aux1 (__`  
`Deque_iterator< _Tp, _Ref, _Ptr > __first1, __Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__equal_aux1 (__Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, __Deque_iterator< _Tp1, _Ref1,`  
`_Ptr1 > __last1, __Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)`
- `template<typename _II, typename _Tp, typename _Ref, typename _Ptr>`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type std::__equal_aux1 (_II`  
`__first1, _II __last1, __Deque_iterator< _Tp, _Ref, _Ptr > __first2)`
- `template<typename _II1, typename _II2>`  
`constexpr bool std::__equal_aux1 (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Flte, typename _Tp>`  
`constexpr void std::__fill_a (_Flte __first, _Flte __last, const _Tp & __value)`
- `template<typename _Lte, typename _Seq, typename _Cat, typename _Tp>`  
`constexpr void std::__fill_a (const __gnu_debug::Safe_iterator< _Lte, _Seq, _Cat > &, const __gnu_debug::Safe_iterator<`  
`_Lte, _Seq, _Cat > &, const _Tp &)`
- `template<typename _Lte, typename _Cont, typename _Tp>`  
`constexpr void std::__fill_a1 (__gnu_cxx::__normal_iterator< _Lte, _Cont > __first, __gnu_cxx::__normal_`  
`iterator< _Lte, _Cont > __last, const _Tp & __value)`
- `constexpr void std::__fill_a1 (__Bit_iterator, __Bit_iterator, const bool &)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::__fill_a1 (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`

- `template<typename _Up, typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_byte< _Up >::__value &&(__are_same< _Up, _Tp >::__value||__`  
`__memcpyable_integer< _Tp >::__width), void >::__type std::__fill_a1 (_Up *__first, _Up *__last, const _Tp`  
`&__x)`
- `template<typename _Tp, typename _VTp>`  
`void std::__fill_a1 (const ::Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::Deque_iterator< _Tp, _Tp`  
`&, _Tp * > &__last, const _VTp &__value)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::output_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::random_access_iterator_tag)`
- `template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>`  
`constexpr ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > std::__fill_n_a (const ::__gnu_debug::Safe_iterator<`  
`_Ite, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator std::__fill_n_a1 (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Iterator, typename _Predicate>`  
`constexpr _Iterator std::__find_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool std::__is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2`  
`__first2, _BinaryPredicate __pred)`
- `template<typename _I1, typename _I2>`  
`constexpr bool std::__lexicographical_compare_aux (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::__lexicographical_compare_aux (_I1, _I1, const ::__gnu_debug::Safe_iterator< _I2,`  
`_Seq2, _Cat2 > &, const ::__gnu_debug::Safe_iterator< _I2, _Seq2, _Cat2 > &)`
- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool std::__lexicographical_compare_aux (const ::__gnu_debug::Safe_iterator< _I1, _Seq1, <`  
`_Cat1 > &, const ::__gnu_debug::Safe_iterator< _I1, _Seq1, _Cat1 > &, _I2, _I2)`
- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::__lexicographical_compare_aux (const ::__gnu_debug::Safe_iterator< _I1, _Seq1, <`  
`_Cat1 > &, const ::__gnu_debug::Safe_iterator< _I1, _Seq1, _Cat1 > &, const ::__gnu_debug::Safe_iterator<`  
`_I2, _Seq2, _Cat2 > &, const ::__gnu_debug::Safe_iterator< _I2, _Seq2, _Cat2 > &)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_<`  
`iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::Deque_iterator<`  
`_Tp2, _Ref2, _Ptr2 > __last2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>`  
`bool std::__lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_<`  
`iterator< _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 *__first2, _Tp2 *__last2)`
- `template<typename _I1, typename _I2>`  
`constexpr bool std::__lexicographical_compare_aux1 (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__lexicographical_compare_aux1 (_Tp1 *__first1, _Tp1 *__last1, ::Deque_iterator< _Tp2, _Ref2,`  
`_Ptr2 > __first2, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)`
- `template<typename _I1, typename _I2, typename _Compare>`  
`constexpr bool std::__lexicographical_compare_impl (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, <`  
`Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp std::__lg (_Tp __n)`

- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp, typename _Up>`  
`constexpr int std::memcmp (const _Tp *__first1, const _Up *__first2, size_t __num)`
- `template<typename _Tp>`  
`constexpr auto std::min_cmp (_Tp __x, _Tp __y)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `constexpr long long std::size_to_integer (double __n)`
- `constexpr long long std::size_to_integer (float __n)`
- `constexpr int std::size_to_integer (int __n)`
- `constexpr long std::size_to_integer (long __n)`
- `constexpr long long std::size_to_integer (long double __n)`
- `constexpr long long std::size_to_integer (long long __n)`
- `constexpr unsigned std::size_to_integer (unsigned __n)`
- `constexpr unsigned long std::size_to_integer (unsigned long __n)`
- `constexpr unsigned long long std::size_to_integer (unsigned long long __n)`
- `template<typename _II, typename _OI>`  
`constexpr _OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2>`  
`constexpr _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _II1, typename _II2>`  
`constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _II2>`  
`constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>`  
`constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>`  
`constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _IIter2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp>`  
`constexpr _OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II1, typename _II2>`  
`constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`

- `template<typename _II1, typename _II2, typename _Compare>`  
`constexpr bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _InputIter1, typename _InputIter2>`  
`constexpr auto std::lexicographical\_compare\_three\_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2, _InputIter2 __last2)`
- `template<typename _InputIter1, typename _InputIter2, typename _Comp>`  
`constexpr auto std::lexicographical\_compare\_three\_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2, _InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _II, typename _OI>`  
`constexpr _OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2>`  
`constexpr _BI2 std::move\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

### 6.149.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 6.150 `std_bvector.h` File Reference

### Classes

- struct `std::hash<::vector< bool, \_Alloc > >`
- class `std::vector< bool, \_Alloc >`

## Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_ALWAYS_INLINE`

## Typedefs

- typedef unsigned long `std::_Bit_type`

## Enumerations

- enum { `_S_word_bit` }

## Functions

- constexpr void `std::__fill_a1` (::\_Bit\_iterator, ::\_Bit\_iterator, const bool &)
- constexpr void `std::__fill_bvector` (\_Bit\_type \*\_\_v, unsigned int \_\_first, unsigned int \_\_last, bool \_\_x) noexcept
- constexpr void `std::__fill_bvector_n` (\_Bit\_type \*, size\_t, bool) noexcept

### 6.150.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 6.151 `std_construct.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_Tp, typename... \_Args>  
constexpr void `std::_Construct` (\_Tp \*\_\_p, \_Args &&... \_\_args)
- template<typename \_T1>  
void `std::_Construct_novalue` (\_T1 \*\_\_p)
- template<typename \_ForwardIterator>  
constexpr void `std::_Destroy` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_Tp>  
constexpr void `std::_Destroy` (\_Tp \*\_\_pointer)
- template<typename \_ForwardIterator, typename \_Size>  
constexpr \_ForwardIterator `std::_Destroy_n` (\_ForwardIterator \_\_first, \_Size \_\_count)
- template<typename \_Tp, typename... \_Args>  
requires (!is\_unbounded\_array\_v<\_Tp>) && requires { ::new((void\*)0) \_Tp(std::declval<\_Args>()...); }  
constexpr \_Tp \* `std::construct_at` (\_Tp \*\_\_location, \_Args &&... \_\_args) noexcept(noexcept(::new((void \*) 0) \_Tp(std::declval<\_Args>()...)))
- template<typename \_ForwardIterator>  
constexpr void `std::destroy` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_Tp>  
constexpr void `std::destroy_at` (\_Tp \*\_\_location)
- template<typename \_ForwardIterator, typename \_Size>  
constexpr \_ForwardIterator `std::destroy_n` (\_ForwardIterator \_\_first, \_Size \_\_count)

### 6.151.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.152 `std_deque.h` File Reference

### Classes

- class `std::_Deque_base<_Tp, _Alloc>`
- struct `std::_Deque_iterator<_Tp, _Ref, _Ptr>`
- class `std::deque<_Tp, _Alloc>`

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_DEQUE_BUF_SIZE`

### Functions

- constexpr `size_t std::__deque_buf_size (size_t __size)`
- template<typename `_InputIterator`, typename `_ValT` = `typename iterator_traits<_InputIterator>::value_type`, typename `_Allocator` = `allocator<_ValT>`, typename = `_RequireInputIter<_InputIterator>`, typename = `_RequireAllocator<_Allocator>`>>  
`std::deque` (`_InputIterator`, `_InputIterator`, `_Allocator=_Allocator()`) -> `deque<_ValT, _Allocator>`
- template<typename `_Tp`, typename `_Alloc`>  
`__detail::__synth3way_t<_Tp>` `std::operator<=>` (`const deque<_Tp, _Alloc>` &`__x`, `const deque<_Tp, _Alloc>` &`__y`)
- template<typename `_Tp`, typename `_Alloc`>  
`bool std::operator==` (`const deque<_Tp, _Alloc>` &`__x`, `const deque<_Tp, _Alloc>` &`__y`)
- template<typename `_Tp`, typename `_Alloc`>  
`void std::swap` (`deque<_Tp, _Alloc>` &`__x`, `deque<_Tp, _Alloc>` &`__y`) `noexcept`(/\**conditional* \*/)

### 6.152.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

### 6.152.2 Macro Definition Documentation

#### `_GLIBCXX_DEQUE_BUF_SIZE`

```
#define _GLIBCXX_DEQUE_BUF_SIZE
```

This function controls the size of memory nodes.

#### Parameters

<code>__size</code>	The size of an element.
---------------------	-------------------------

### Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

## 6.153 `std_function.h` File Reference

### Classes

- struct `std::binary_function<_Arg1, _Arg2, _Result >`
- class `std::binary_negate<_Predicate >`
- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg >`
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg >`
- class `std::const_mem_fun_ref_t<_Ret, _Tp >`
- class `std::const_mem_fun_t<_Ret, _Tp >`
- struct `std::divides<_Tp >`
- struct `std::divides<void >`
- struct `std::equal_to<_Tp >`
- struct `std::equal_to<void >`
- struct `std::greater<_Tp >`
- struct `std::greater<void >`
- struct `std::greater_equal<_Tp >`
- struct `std::greater_equal<void >`
- struct `std::less<_Tp >`
- struct `std::less<void >`
- struct `std::less_equal<_Tp >`
- struct `std::less_equal<void >`
- struct `std::logical_and<_Tp >`
- struct `std::logical_and<void >`
- struct `std::logical_not<_Tp >`
- struct `std::logical_not<void >`
- struct `std::logical_or<_Tp >`
- struct `std::logical_or<void >`
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg >`
- class `std::mem_fun1_t<_Ret, _Tp, _Arg >`
- class `std::mem_fun_ref_t<_Ret, _Tp >`
- class `std::mem_fun_t<_Ret, _Tp >`
- struct `std::minus<_Tp >`
- struct `std::minus<void >`
- struct `std::modulus<_Tp >`
- struct `std::modulus<void >`
- struct `std::multiplies<_Tp >`
- struct `std::multiplies<void >`
- struct `std::negate<_Tp >`
- struct `std::negate<void >`
- struct `std::not_equal_to<_Tp >`
- struct `std::not_equal_to<void >`
- struct `std::plus<_Tp >`
- class `std::pointer_to_binary_function<_Arg1, _Arg2, _Result >`
- class `std::pointer_to_unary_function<_Arg, _Result >`
- struct `std::unary_function<_Arg, _Result >`
- class `std::unary_negate<_Predicate >`

## Namespaces

- namespace `std`

## Typedefs

- `template<typename _Func, typename _SfinaeType>`  
using `std::__has_is_transparent_t`

## Functions

- `template<typename _Ret, typename _Tp>`  
`const_mem_fun_t<_Ret, _Tp>` `std::mem_fun` (`_Ret(_Tp::*__f)()` const)
- `template<typename _Ret, typename _Tp>`  
`mem_fun_t<_Ret, _Tp>` `std::mem_fun` (`_Ret(_Tp::*__f)()`)
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`const_mem_fun1_t<_Ret, _Tp, _Arg>` `std::mem_fun` (`_Ret(_Tp::*__f)(_Arg)` const)
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_t<_Ret, _Tp, _Arg>` `std::mem_fun` (`_Ret(_Tp::*__f)(_Arg)`)
- `template<typename _Ret, typename _Tp>`  
`const_mem_fun_ref_t<_Ret, _Tp>` `std::mem_fun_ref` (`_Ret(_Tp::*__f)()` const)
- `template<typename _Ret, typename _Tp>`  
`mem_fun_ref_t<_Ret, _Tp>` `std::mem_fun_ref` (`_Ret(_Tp::*__f)()`)
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`const_mem_fun1_ref_t<_Ret, _Tp, _Arg>` `std::mem_fun_ref` (`_Ret(_Tp::*__f)(_Arg)` const)
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_ref_t<_Ret, _Tp, _Arg>` `std::mem_fun_ref` (`_Ret(_Tp::*__f)(_Arg)`)
- `template<typename _Predicate>`  
constexpr `unary_negate<_Predicate>` `std::not1` (const `_Predicate &__pred`)
- `template<typename _Predicate>`  
constexpr `binary_negate<_Predicate>` `std::not2` (const `_Predicate &__pred`)
- `template<typename _Arg, typename _Result>`  
`pointer_to_unary_function<_Arg, _Result>` `std::ptr_fun` (`_Result(*__x)(_Arg)`)
- `template<typename _Arg1, typename _Arg2, typename _Result>`  
`pointer_to_binary_function<_Arg1, _Arg2, _Result>` `std::ptr_fun` (`_Result(*__x)(_Arg1, _Arg2)`)

### 6.153.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.154 `std_heap.h` File Reference

### Namespaces

- namespace `std`

### Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>`  
constexpr void `std::__adjust_heap` (`_RandomAccessIterator __first`, `_Distance __holeIndex`, `_Distance __len`, `_Tp __value`, `_Compare __comp`)
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance>`  
constexpr bool `std::__is_heap` (`_RandomAccessIterator __first`, `_Compare __comp`, `_Distance __n`)



- `template<typename _RandomAccessIterator, typename _Distance>`  
`constexpr bool std::is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator>`  
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare>`  
`constexpr _Distance std::is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>`  
`constexpr void std::push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

### 6.154.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

## 6.155 bits/stl\_iterator.h File Reference

### Classes

- class [std::back\\_insert\\_iterator<\\_Container>](#)
- class [std::common\\_iterator<\\_It, \\_Sent>](#)
- class [std::counted\\_iterator<\\_It>](#)
- class [std::front\\_insert\\_iterator<\\_Container>](#)
- class [std::insert\\_iterator<\\_Container>](#)
- class [std::move\\_iterator<\\_Iterator>](#)
- class [std::move\\_sentinel<\\_Sent>](#)
- class [std::reverse\\_iterator<\\_Iterator>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)

### Macros

- `#define GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR(_Iter)`
- `#define GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

### Typedefs

- `template<typename _InputIterator>`  
`using std::__iter_key_t`
- `template<typename _InputIterator>`  
`using std::__iter_to_alloc_t`
- `template<typename _InputIterator>`  
`using std::__iter_val_t`

### Functions

- `template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>>`  
`constexpr _ReturnType std::__make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>>>`  
`constexpr _ReturnType std::__make_move_if_noexcept_iterator (_Tp *__i)`
- `template<typename _Iterator>`  
`constexpr reverse_iterator<_Iterator> std::__make_reverse_iterator (_Iterator __i)`
- `template<typename _Container>`  
`constexpr back_insert_iterator<_Container> std::back_inserter (_Container &__x)`
- `template<typename _Container>`  
`constexpr front_insert_iterator<_Container> std::front_inserter (_Container &__x)`
- `template<typename _Container>`  
`constexpr insert_iterator<_Container> std::inserter (_Container &__x, std::__detail::__range_iter_t<_Container> __i)`

- `template<typename _Iterator>`  
`constexpr move\_iterator< _Iterator > std::make_move_iterator ( _Iterator __i)`
- `template<typename _Iterator>`  
`constexpr reverse\_iterator< _Iterator > std::make_reverse_iterator ( _Iterator __i)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator!= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator>`  
`requires requires { { __x.base() + __n } -> same_as< _Iterator>; }`  
`constexpr move\_iterator< _Iterator > std::operator+ (typename move\_iterator< _Iterator >::difference_type __n, const move\_iterator< _Iterator > &__x)`
- `template<typename _Iterator>`  
`constexpr reverse\_iterator< _Iterator > std::operator+ (typename reverse\_iterator< _Iterator >::difference_type __n, const reverse\_iterator< _Iterator > &__x)`
- `template<typename _IteratorL, typename _IteratorR>`  
`constexpr auto std::operator- (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y) -> decltype(__x.base() - __y.base())`
- `template<typename _IteratorL, typename _IteratorR>`  
`constexpr auto std::operator- (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y) -> decltype(__y.base() - __x.base())`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator< (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator<= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator>= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare\_three\_way\_result\_t< _Iterator > std::operator<= (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare\_three\_way\_result\_t< _IteratorL, _IteratorR > std::operator<= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare\_three\_way\_result\_t< _Iterator, _Iterator > std::operator<= (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare\_three\_way\_result\_t< _IteratorL, _IteratorR > std::operator<= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator>`  
`constexpr bool std::operator== (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`

```
constexpr bool std::operator== (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR >
&__y)
• template<typename _Iterator>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator== (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator >
&__y)
• template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator== (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR >
&__y)
• template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
constexpr bool std::operator> (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR >
&__y)
• template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR
> &__y)
• template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator>= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR >
&__y)
• template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator>= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR
> &__y)
```

## Variables

```
• template<typename _Iterator1, typename _Iterator2>
  constexpr bool std::disable_sized_sentinel_for< move\_iterator< _Iterator1 >, move\_iterator< _Iterator2
> >
• template<typename _Iterator1, typename _Iterator2>
  constexpr bool std::disable_sized_sentinel_for< reverse\_iterator< _Iterator1 >, reverse\_iterator< _↵
Iterator2 > >
```

### 6.155.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

## 6.156 debug/stl\_iterator.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

```
• template<typename _Iterator>
  constexpr auto __gnu_debug::__base (const std::move\_iterator< _Iterator > &__it) -> decltype(std::make\_↵
move\_iterator(__base(__it.base())))
```

- `template<typename _Iterator, typename _Sequence>`  
`constexpr std::reverse_iterator< _Iterator > __gnu_debug::__base (const std::reverse_iterator< _Safe_iterator<`  
`_Iterator, _Sequence, std::random_access_iterator_tag > > &__it)`
- `template<typename _Iterator, typename _Size>`  
`constexpr bool __gnu_debug::__can_advance (const std::move_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator, typename _Diff>`  
`constexpr bool __gnu_debug::__can_advance (const std::move_iterator< _Iterator > &__it, const std::pair<`  
`_Diff, _Distance_precision > &__dist, int __way)`
- `template<typename _Iterator, typename _Size>`  
`constexpr bool __gnu_debug::__can_advance (const std::reverse_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator, typename _Diff>`  
`constexpr bool __gnu_debug::__can_advance (const std::reverse_iterator< _Iterator > &__it, const std::pair<`  
`_Diff, _Distance_precision > &__dist, int __way)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (const std::move_iterator< ↵`  
`_Iterator > &__first, const std::move_iterator< _Iterator > &__last)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (const std::reverse_iterator<`  
`_Iterator > &__first, const std::reverse_iterator< _Iterator > &__last)`
- `template<typename _Iterator>`  
`constexpr auto __gnu_debug::__unsafe (const std::move_iterator< _Iterator > &__it) -> decltype(std::make↵`  
`_move_iterator(__unsafe(__it.base())))`
- `template<typename _Iterator>`  
`constexpr auto __gnu_debug::__unsafe (const std::reverse_iterator< _Iterator > &__it) -> decltype(std::↵`  
`make_reverse_iterator(__unsafe(__it.base())))`
- `template<typename _Iterator>`  
`constexpr bool __gnu_debug::__valid_range (const std::move_iterator< _Iterator > &__first, const`  
`std::move_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _Iterator>`  
`constexpr bool __gnu_debug::__valid_range (const std::reverse_iterator< _Iterator > &__first, const`  
`std::reverse_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`

### 6.156.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.157 `std_iterator_base_funcs.h` File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Functions

- `template<typename _BidirectionalIterator, typename _Distance>`  
`constexpr void std::__advance (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)`
- `template<typename _InputIterator, typename _Distance>`  
`constexpr void std::__advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _OutputIterator, typename _Distance>`  
`void std::__advance (_OutputIterator &, _Distance, output_iterator_tag)=delete`
- `template<typename _RandomAccessIterator, typename _Distance>`  
`constexpr void std::__advance (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)`

- `template<typename _Tp>`  
`ptrdiff_t std::__distance (::_List_const_iterator< _Tp >, ::_List_const_iterator< _Tp >, input\_iterator\_tag)`
- `template<typename _Tp>`  
`ptrdiff_t std::__distance (::_List_iterator< _Tp >, ::_List_iterator< _Tp >, input\_iterator\_tag)`
- `template<typename _InputIterator>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::__distance (_InputIterator __first, _InputIterator __last, input\_iterator\_tag)`
- `template<typename _OutputIterator>`  
`void std::__distance (_OutputIterator, _OutputIterator, output\_iterator\_tag)=delete`
- `template<typename _RandomAccessIterator>`  
`constexpr iterator\_traits< _RandomAccessIterator >::difference_type std::__distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Distance>`  
`constexpr void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`constexpr _InputIterator std::next (_InputIterator __x, typename iterator\_traits< _InputIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator>`  
`constexpr _BidirectionalIterator std::prev (_BidirectionalIterator __x, typename iterator\_traits< _BidirectionalIterator >::difference_type __n=1)`

### 6.157.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility functions, such as `distance()` and `advance()`.

## 6.158 `std_iterator_base_types.h` File Reference

### Classes

- struct [std::bidirectional\\_iterator\\_tag](#)
- struct [std::contiguous\\_iterator\\_tag](#)
- struct [std::forward\\_iterator\\_tag](#)
- struct [std::input\\_iterator\\_tag](#)
- struct [std::iterator< \\_Category, \\_Tp, \\_Distance, \\_Pointer, \\_Reference >](#)
- struct [std::iterator\\_traits< \\_Iterator >](#)
- struct [std::iterator\\_traits< \\_Tp \\* >](#)
- struct [std::output\\_iterator\\_tag](#)
- struct [std::random\\_access\\_iterator\\_tag](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define \_GLIBCXX26\_ALGO\_DEF\_VAL\_T(_Iterator)`
- `#define \_GLIBCXX26\_DEF\_VAL\_T(T)`

## Typedefs

- `template<typename _Iter>`  
using `std::__iter_category_t`
- `template<typename _InIter>`  
using `std::__RequireInputIter`

## Functions

- `template<typename _Iter>`  
constexpr `iterator_traits<_Iter>::iterator_category` `std::__iterator_category` (const `_Iter` &)

### 6.158.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

## 6.159 `std::list.h` File Reference

### Classes

- class `std::_List_base<_Tp, _Alloc>`
- struct `std::_List_const_iterator<_Tp>`
- struct `std::_List_iterator<_Tp>`
- struct `std::_List_node<_Tp>`
- struct `std::__detail::_List_node_base`
- struct `std::__detail::_List_node_header`
- class `std::list<_Tp, _Alloc>`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Macros

- `#define _GLIBCXX_LIST_REMOVE_RETURN_TYPE_TAG`
- `#define _GLIBCXX_USE_ALLOC_PTR_FOR_LIST`

### Functions

- `template<typename _Tp>`  
`ptrdiff_t` `std::__distance` (`::_List_const_iterator<_Tp>`, `::_List_const_iterator<_Tp>`, `input_iterator_tag`)
- `template<typename _Tp>`  
`ptrdiff_t` `std::__distance` (`::_List_iterator<_Tp>`, `::_List_iterator<_Tp>`, `input_iterator_tag`)
- `template<bool _Const, typename _Ptr>`  
`ptrdiff_t` `std::__distance` (`__list::iterator<_Const, _Ptr>` `__first`, `__list::iterator<_Const, _Ptr>` `__last`, `input_iterator_tag` `__tag`)
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::list` (`_InputIterator`, `_InputIterator`, `_Allocator=_Allocator()`) -> `list<_ValT, _Allocator>`
- `template<typename _Tp, typename _Alloc>`  
`__detail::__synth3way_t<_Tp>` `std::operator<=>` (const `list<_Tp, _Alloc>` &`__x`, const `list<_Tp, _Alloc>` &`__y`)

- `template<typename _Tp, typename _Alloc>`  
`bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)`

### 6.159.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

## 6.160 `std_map.h` File Reference

### Classes

- class [std::map](#)< [\\_Key](#), [\\_Tp](#), [\\_Compare](#), [\\_Alloc](#) >

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::map (_InputIterator, _InputIterator, _Allocator) -> map< \_\_iter\_key\_t< _InputIterator >, \_\_iter\_val\_t< _InputIterator >, less< \_\_iter\_key\_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less< \_\_iter\_key\_t< _InputIterator > >, typename _Allocator = allocator< \_\_iter\_val\_t< _InputIterator > >, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::map (_InputIterator, _InputIterator, _Compare=\_Compare(), _Allocator=\_Allocator()) -> map< \_\_iter\_key\_t< _InputIterator >, \_\_iter\_val\_t< _InputIterator >, \_Compare, \_Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`std::map (initializer\_list< pair< \_Key, \_Tp > >, _Allocator) -> map< \_Key, \_Tp, less< \_Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Compare = less< \_Key >, typename _Allocator = allocator< pair< const \_Key, \_Tp > >, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::map (initializer\_list< pair< \_Key, \_Tp > >, _Compare=\_Compare(), _Allocator=\_Allocator()) -> map< \_Key, \_Tp, \_Compare, \_Allocator >`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`\_\_detail::\_\_synth3way\_t< pair< const \_Key, \_Tp > > std::operator<=> (const map< \_Key, \_Tp, \_Compare, \_Alloc > &__x, const map< \_Key, \_Tp, \_Compare, \_Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator== (const map< \_Key, \_Tp, \_Compare, \_Alloc > &__x, const map< \_Key, \_Tp, \_Compare, \_Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`void std::swap (map< \_Key, \_Tp, \_Compare, \_Alloc > &__x, map< \_Key, \_Tp, \_Compare, \_Alloc > &__y) noexcept(/*conditional */)`

### 6.160.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.



## 6.161 std\_multimap.h File Reference

### Classes

- class [std::multimap<\\_Key, \\_Tp, \\_Compare, \\_Alloc>](#)

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>  
**std::multimap** (\_InputIterator, \_InputIterator, \_Allocator) -> multimap<\_\_iter\_key\_t<\_InputIterator>, \_\_iter\_val\_t<\_InputIterator>, [less](#)<\_\_iter\_key\_t<\_InputIterator>, \_\_iter\_val\_t<\_InputIterator>, \_Allocator>>
- template<typename \_InputIterator, typename \_Compare = [less](#)<\_\_iter\_key\_t<\_InputIterator>, \_\_iter\_val\_t<\_InputIterator>, \_Allocator = allocator<\_\_iter\_key\_t<\_InputIterator>, \_\_iter\_val\_t<\_InputIterator>>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireAllocator<\_Allocator>>>  
**std::multimap** (\_InputIterator, \_InputIterator, \_Compare=\_Compare(), \_Allocator=\_Allocator()) -> multimap<\_\_iter\_key\_t<\_InputIterator>, \_\_iter\_val\_t<\_InputIterator>, \_Compare, \_Allocator>
- template<typename \_Key, typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>>  
**std::multimap** ([initializer\\_list](#)< [pair](#)<\_Key, \_Tp>, \_Allocator) -> multimap<\_Key, \_Tp, [less](#)<\_Key>, \_Allocator>
- template<typename \_Key, typename \_Tp, typename \_Compare = [less](#)<\_Key>, typename \_Allocator = allocator<[pair](#)<const \_Key, \_Tp>>, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireAllocator<\_Allocator>>>  
**std::multimap** ([initializer\\_list](#)< [pair](#)<\_Key, \_Tp>, \_Compare=\_Compare(), \_Allocator=\_Allocator()) -> multimap<\_Key, \_Tp, \_Compare, \_Allocator>
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Alloc>  
bool [std::operator!=](#) (const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_x, const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Alloc>  
bool [std::operator<](#) (const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_x, const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Alloc>  
bool [std::operator<=](#) (const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_x, const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Alloc>  
bool [std::operator==](#) (const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_x, const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Alloc>  
bool [std::operator>](#) (const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_x, const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Alloc>  
bool [std::operator>=](#) (const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_x, const [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Alloc>  
void [std::swap](#) ([multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_x, [multimap](#)<\_Key, \_Tp, \_Compare, \_Alloc> &\_\_y) noexcept([/\\*conditional \\*/](#))

#### 6.161.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

## 6.162 std\_multiset.h File Reference

### Classes

- class [std::multiset<\\_Key, \\_Compare, \\_Alloc>](#)

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>  
**std::multiset** (\_InputIterator, \_InputIterator, \_Allocator) -> multiset< typename [iterator\\_traits](#)<\_InputIterator>::value\_type, [less](#)< typename [iterator\\_traits](#)<\_InputIterator>::value\_type >, \_Allocator >
- template<typename \_InputIterator, typename \_Compare = less<typename iterator\_traits<\_InputIterator>::value\_type>, typename \_Allocator = allocator<typename iterator\_traits<\_InputIterator>::value\_type>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireAllocator<\_Allocator>>  
**std::multiset** (\_InputIterator, \_InputIterator, \_Compare=\_Compare(), \_Allocator=\_Allocator()) -> multiset< typename [iterator\\_traits](#)<\_InputIterator>::value\_type, \_Compare, \_Allocator >
- template<typename \_Key, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>  
**std::multiset** ([initializer\\_list](#)<\_Key>, \_Allocator) -> multiset< \_Key, [less](#)<\_Key>, \_Allocator >
- template<typename \_Key, typename \_Compare = less<\_Key>, typename \_Allocator = allocator<\_Key>, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireAllocator<\_Allocator>>  
**std::multiset** ([initializer\\_list](#)<\_Key>, \_Compare=\_Compare(), \_Allocator=\_Allocator()) -> multiset< \_Key, \_Compare, \_Allocator >
- template<typename \_Key, typename \_Compare, typename \_Alloc>  
bool **std::operator!=** (const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_x, const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Compare, typename \_Alloc>  
bool **std::operator<** (const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_x, const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Compare, typename \_Alloc>  
bool **std::operator<=** (const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_x, const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Compare, typename \_Alloc>  
bool **std::operator==** (const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_x, const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Compare, typename \_Alloc>  
bool **std::operator>** (const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_x, const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Compare, typename \_Alloc>  
bool **std::operator>=** (const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_x, const [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_y)
- template<typename \_Key, typename \_Compare, typename \_Alloc>  
void **std::swap** ([multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_x, [multiset](#)<\_Key, \_Compare, \_Alloc> &\_\_y) noexcept(*/\*conditional \*/*)

#### 6.162.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

## 6.163 `std_numeric.h` File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_MOVE_IF_20(_E)`

### Functions

- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __↔  
binary_op)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator ↔  
__result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator ↔  
__result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp  
__init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _Binary↔  
Operation2>  
constexpr _Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp  
__init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result,  
_BinaryOperation __binary_op)`

#### 6.163.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

## 6.164 `std_pair.h` File Reference

### Classes

- struct `std::pair< _T1, _T2 >`
- struct `std::piecewise\_construct\_t`
- struct `std::tuple\_element< 0, pair< _Tp1, _Tp2 > >`
- struct `std::tuple\_element< 1, pair< _Tp1, _Tp2 > >`
- struct `std::tuple\_size< pair< _Tp1, _Tp2 > >`

### Namespaces

- namespace `std`

## Functions

- `template<typename _T1, typename _T2>`  
`constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type`  
`> std::make_pair ( _T1 &&__x, _T2 &&__y)`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp && std::get (const pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp & std::get (const pair< _Tp, _Up > &__p) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (const pair< _Tp1, _Tp2 > &&__`  
`__in) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (const pair< _Tp1, _Tp2 > &__in)`  
`noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp && std::get (const pair< _Up, _Tp > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp & std::get (const pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp && std::get (pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp & std::get (pair< _Tp, _Up > &__p) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (pair< _Tp1, _Tp2 > &&__in) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp && std::get (pair< _Up, _Tp > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp & std::get (pair< _Up, _Tp > &__p) noexcept`

## Variables

- `template<typename _Tp>`  
`constexpr bool std::__is_pair`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::__is_pair< pair< _Tp, _Up > >`
- `constexpr piecewise_construct_t std::piecewise_construct`
- `template<typename _Tp1, typename _Tp2>`  
`constexpr size_t std::tuple_size_v< const pair< _Tp1, _Tp2 > >`
- `template<typename _Tp1, typename _Tp2>`  
`constexpr size_t std::tuple_size_v< pair< _Tp1, _Tp2 > >`

### 6.164.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

## 6.165 `std::queue.h` File Reference

### Classes

- class `std::priority_queue< _Tp, _Sequence, _Compare >`
- class `std::queue< _Tp, _Sequence >`

## Namespaces

- namespace `std`

## Functions

- `template<typename _Tp, typename _Seq>`  
`bool std::operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, three_way_comparable _Seq>`  
`compare_three_way_result_t< _Seq > std::operator<=> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Compare, typename _Container, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>>`  
`std::priority_queue (_Compare, _Container) -> priority_queue< typename _Container::value_type, _Container, _Compare >`
- `template<typename _Compare, typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>>`  
`std::priority_queue (_Compare, _Container, _Allocator) -> priority_queue< typename _Container::value_type, _Container, _Compare >`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Compare = less<_ValT>, typename _Container = vector<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>>`  
`std::priority_queue (_InputIterator, _InputIterator, _Compare=_Compare(), _Container=_Container()) -> priority_queue< _ValT, _Container, _Compare >`
- `template<typename _Container, typename = _RequireNotAllocator<_Container>>>`  
`std::queue (_Container) -> queue< typename _Container::value_type, _Container >`
- `template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>>`  
`std::queue (_Container, _Allocator) -> queue< typename _Container::value_type, _Container >`
- `template<typename _Tp, typename _Sequence, typename _Compare>`  
`enable_if< __and< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::type std::swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Seq>`  
`enable_if< __is_swappable< _Seq >::value >::type std::swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`

### 6.165.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

## 6.166 `std_raw_storage_iter.h` File Reference

### Classes

- class `std::raw_storage_iterator<_OutputIterator, _Tp>`

### Namespaces

- namespace `std`

#### 6.166.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.167 `std_relops.h` File Reference

### Namespaces

- namespace `std`
- namespace `std::rel_ops`

### Functions

- `template<class _Tp>`  
`bool std::rel_ops::operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`  
`bool std::rel_ops::operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`  
`bool std::rel_ops::operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`  
`bool std::rel_ops::operator>= (const _Tp &__x, const _Tp &__y)`

#### 6.167.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

This file is only included by `<utility>`, which is required by the standard to define namespace `rel_ops` and its contents.

## 6.168 `std_set.h` File Reference

### Classes

- class `std::set<_Key, _Compare, _Alloc>`

### Namespaces

- namespace `std`

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator!= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator< (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`

- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits<_InputIterator>::value_type, less< typename iterator_traits<_InputIterator>::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`std::set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits<_InputIterator>::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`std::set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

### 6.168.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

## 6.169 std::stack.h File Reference

### Classes

- class `std::stack< _Tp, _Sequence >`

### Namespaces

- namespace `std`

### Functions

- `template<typename _Tp, typename _Seq>`  
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, three_way_comparable _Seq>`  
`compare_three_way_result_t< _Seq > std::operator<=> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`

- `template<typename _Tp, typename _Seq>`  
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Container, typename = _RequireNotAllocator<_Container>>`  
`std::stack ( _Container) -> stack< typename _Container::value_type, _Container >`
- `template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>`  
`std::stack ( _Container, _Allocator) -> stack< typename _Container::value_type, _Container >`
- `template<typename _Tp, typename _Seq>`  
`enable_if< __is_swappable< _Seq >::value >::type std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`

### 6.169.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stack>`.

## 6.170 `std_tempbuf.h` File Reference

### Classes

- class `std::_Temporary_buffer< _ForwardIterator, _Tp >`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Macros

- `#define _GLIBCXX_OPERATOR_DELETE`
- `#define _GLIBCXX_OPERATOR_NEW`
- `#define _GLIBCXX_SIZED_DEALLOC(T, p, n)`

### Functions

- `template<typename _Tp>`  
`_Tp * std::__detail::__get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _Tp>`  
`void std::__detail::__return_temporary_buffer ( _Tp * __p, size_t __len)`
- `template<typename _Tp, typename _ForwardIterator>`  
`void std::__uninitialized_construct_buf ( _Tp * __first, _Tp * __last, _ForwardIterator __seed)`
- `template<typename _Tp>`  
`pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _Tp>`  
`void std::return_temporary_buffer ( _Tp * __p)`

### 6.170.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.



## 6.171 `std_tree.h` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_USE_ALLOC_PTR_FOR_RB_TREE`

### Enumerations

- enum `_Rb_tree_color` { `_S_red`, `_S_black` }

### Functions

- unsigned int `std::_Rb_tree_black_count` (const `_Rb_tree_node_base` \* \_\_node, const `_Rb_tree_node_base` \* \_\_root) throw ()
- `_Rb_tree_node_base` \* `std::_Rb_tree_decrement` (`_Rb_tree_node_base` \* \_\_x) throw ()
- `_Rb_tree_node_base` \* `std::_Rb_tree_increment` (`_Rb_tree_node_base` \* \_\_x) throw ()
- void `std::_Rb_tree_insert_and_rebalance` (const bool \_\_insert\_left, `_Rb_tree_node_base` \* \_\_x, `_Rb_tree_node_base` \* \_\_p, `_Rb_tree_node_base` & \_\_header) throw ()
- `_Rb_tree_node_base` \* `std::_Rb_tree_rebalance_for_erase` (`_Rb_tree_node_base` \* const \_\_z, `_Rb_tree_node_base` & \_\_header) throw ()
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc`>  
void `std::swap` (`_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & \_\_x, `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > & \_\_y)

#### 6.171.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>` or `<set>`.

## 6.172 `std_uninitialized.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- template<typename `_InputIterator`, typename `_ForwardIterator`>  
`_ForwardIterator` `std::uninitialized_copy` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `_ForwardIterator` \_\_result)
- template<typename `_InputIterator`, typename `_Size`, typename `_ForwardIterator`>  
`_ForwardIterator` `std::uninitialized_copy_n` (`_InputIterator` \_\_first, `_Size` \_\_n, `_ForwardIterator` \_\_result)
- template<typename `_ForwardIterator`>  
void `std::uninitialized_default_construct` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last)
- template<typename `_ForwardIterator`, typename `_Size`>  
`_ForwardIterator` `std::uninitialized_default_construct_n` (`_ForwardIterator` \_\_first, `_Size` \_\_count)
- template<typename `_ForwardIterator`, typename `_Tp`>  
void `std::uninitialized_fill` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, const `_Tp` & \_\_x)
- template<typename `_ForwardIterator`, typename `_Size`, typename `_Tp`>  
`_ForwardIterator` `std::uninitialized_fill_n` (`_ForwardIterator` \_\_first, `_Size` \_\_n, const `_Tp` & \_\_x)
- template<typename `_InputIterator`, typename `_ForwardIterator`>  
`_ForwardIterator` `std::uninitialized_move` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `_ForwardIterator` \_\_result)

- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>`  
`pair< _InputIterator, _ForwardIterator > std::uninitialized_move_n` (`_InputIterator __first`, `_Size __count`, `_ForwardIterator __result`)
- `template<typename _ForwardIterator>`  
`void std::uninitialized_value_construct` (`_ForwardIterator __first`, `_ForwardIterator __last`)
- `template<typename _ForwardIterator, typename _Size>`  
`_ForwardIterator std::uninitialized_value_construct_n` (`_ForwardIterator __first`, `_Size __count`)

### 6.172.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.173 `std_vector.h` File Reference

### Classes

- struct `std::_Vector_base< _Tp, _Alloc >`
- class `std::vector< _Tp, _Alloc >`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Macros

- `#define _GLIBCXX_ASAN_ANNOTATE_BEFORE_DEALLOC`
- `#define _GLIBCXX_ASAN_ANNOTATE_GREW(n)`
- `#define _GLIBCXX_ASAN_ANNOTATE_GROW(n)`
- `#define _GLIBCXX_ASAN_ANNOTATE_REINIT`
- `#define _GLIBCXX_ASAN_ANNOTATE_SHRINK(n)`

### Functions

- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp > std::operator<=>` (`const vector< _Tp, _Alloc > &__x`, `const vector< _Tp, _Alloc > &__y`)
- `template<typename _Tp, typename _Alloc>`  
`constexpr bool std::operator==` (`const vector< _Tp, _Alloc > &__x`, `const vector< _Tp, _Alloc > &__y`)
- `template<typename _Tp, typename _Alloc>`  
`constexpr void std::swap` (`vector< _Tp, _Alloc > &__x`, `vector< _Tp, _Alloc > &__y`) `noexcept` (`/*conditional */`)
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::vector` (`_InputIterator`, `_InputIterator`, `_Allocator=_Allocator()`) `-> vector<_ValT, _Allocator>`

### 6.173.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 6.174 `stream_iterator.h` File Reference

### Classes

- class [std::istream\\_iterator](#)< [\\_Tp](#), [\\_CharT](#), [\\_Traits](#), [\\_Dist](#) >
- class [std::ostream\\_iterator](#)< [\\_Tp](#), [\\_CharT](#), [\\_Traits](#) >

### Namespaces

- namespace [std](#)

#### 6.174.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.175 `streambuf.tcc` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _STREAMBUF_TCC`

### Functions

- `template<typename _CharT, typename _Traits>`  
[streamsize](#) [std::\\_\\_copy\\_streambufs](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_sbin](#), [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_sbout](#))
- `template<typename _CharT, typename _Traits>`  
[streamsize](#) [std::\\_\\_copy\\_streambufs\\_eof](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_sbin](#), [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_sbout](#), `bool` &[\\_\\_ineof](#))

#### 6.175.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<streambuf>`.

## 6.176 `streambuf_iterator.h` File Reference

### Classes

- class [std::istreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) >
- class [std::ostreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) >

### Namespaces

- namespace [std](#)

### Functions

- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if< \_\_is\_char< \_CharT >::__value, ostreambuf\_iterator< \_CharT > >::__type` [std::\\_\\_copy\\_move\\_a2](#) ([\\_CharT](#) \*[\\_\\_first](#), [\\_CharT](#) \*[\\_\\_last](#), [ostreambuf\\_iterator](#)< [\\_CharT](#) > [\\_\\_result](#))

- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__↔`  
`copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_n_a (istreambuf_iterator<`  
`_CharT > __it, _Size __n, _CharT * __result, bool __strict)`
- `template<typename _CharT, typename _Distance>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type std::advance (istreambuf_iterator<`  
`_CharT > & __i, _Distance __n)`
- `template<typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >`  
`__result)`
- `template<typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::find`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT & __val)`
- `template<typename _CharT, typename _Traits>`  
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT,`  
`_Traits > & __b)`

### 6.176.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.177 bits/string\_view.tcc File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_STRING_VIEW_TCC`

### 6.177.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string_view>`.

## 6.178 experimental/bits/string\_view.tcc File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_STRING_VIEW_TCC`

### 6.178.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/string_view>`.

## 6.179 stringfwd.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef [basic\\_string](#)< char > [std::string](#)
- typedef [basic\\_string](#)< char16\_t > [std::u16string](#)
- typedef [basic\\_string](#)< char32\_t > [std::u32string](#)
- typedef [basic\\_string](#)< wchar\_t > [std::wstring](#)

### 6.179.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 6.180 text\_encoding-data.h File Reference

### Macros

- `#define _GLIBCXX_TEXT_ENCODING_UTF8_OFFSET`

### 6.180.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<text_encoding>`.

## 6.181 this\_thread\_sleep.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::this\\_thread](#)

### Functions

- `template<typename _Rep, typename _Period>`  
`void std::this\_thread::sleep\_for (const chrono::duration< _Rep, _Period > &__rtime)`
- `template<typename _Clock, typename _Duration>`  
`void std::this\_thread::sleep\_until (const chrono::time\_point< _Clock, _Duration > &__atime)`

### 6.181.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<thread>`.

## 6.182 unicode-data.h File Reference

### Enumerations

- enum class `_Gcb_property` {  
`_Gcb_Other` , `_Gcb_Control` , `_Gcb_LF` , `_Gcb_CR` ,  
`_Gcb_Extend` , `_Gcb_Prepend` , `_Gcb_SpacingMark` , `_Gcb_L` ,  
`_Gcb_V` , `_Gcb_T` , `_Gcb_ZWJ` , `_Gcb_LV` ,  
`_Gcb_LVT` , `_Gcb_Regional_Indicator` }
- enum class `_InCB` { `_Consonant` , `_Extend` }

### Variables

- constexpr uint32\_t `__escape_edges` []
- constexpr uint32\_t `__gcb_edges` []
- constexpr int `__gcb_shift_bits`
- constexpr uint32\_t `__incb_edges` []
- constexpr char32\_t `__incb_linkers` []
- constexpr char32\_t `__width_edges` []
- constexpr char32\_t `__xpicto_edges` []

### 6.182.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<format>`.

## 6.183 unicode.h File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_GET_UNICODE_DATA`

### Typedefs

- `template<typename _View>`  
`using std::__unicode::Utf16_view`
- `template<typename _View>`  
`using std::__unicode::Utf32_view`
- `template<typename _View>`  
`using std::__unicode::Utf8_view`

### Enumerations

- enum class `_Gcb_property` {  
`_Gcb_Other` , `_Gcb_Control` , `_Gcb_LF` , `_Gcb_CR` ,  
`_Gcb_Extend` , `_Gcb_Prepend` , `_Gcb_SpacingMark` , `_Gcb_L` ,  
`_Gcb_V` , `_Gcb_T` , `_Gcb_ZWJ` , `_Gcb_LV` ,  
`_Gcb_LVT` , `_Gcb_Regional_Indicator` }
- enum class `_InCB` { `_Consonant` , `_Extend` }

## Functions

- constexpr bool **std::\_\_unicode::\_\_charset\_alias\_match** ([string\\_view](#) \_\_a, [string\\_view](#) \_\_b)
- template<typename [\\_CharT](#)>  
constexpr size\_t **std::\_\_unicode::\_\_field\_width** ([basic\\_string\\_view](#)< [\\_CharT](#) > \_\_s)
- constexpr int **std::\_\_unicode::\_\_field\_width** (char32\_t \_\_c) noexcept
- constexpr [\\_Gcb\\_property](#) **std::\_\_unicode::\_\_grapheme\_cluster\_break\_property** (char32\_t \_\_c) noexcept
- constexpr [\\_InCB](#) **std::\_\_unicode::\_\_incb\_property** (char32\_t \_\_c) noexcept
- constexpr bool **std::\_\_unicode::\_\_is\_extended\_pictographic** (char32\_t \_\_c)
- constexpr bool **std::\_\_unicode::\_\_is\_incb\_linker** (char32\_t \_\_c) noexcept
- constexpr bool **std::\_\_unicode::\_\_is\_scalar\_value** (char32\_t \_\_c)
- template<typename [\\_CharT](#)>  
constexpr bool **std::\_\_unicode::\_\_is\_single\_code\_unit** (char32\_t \_\_c)
- constexpr bool **std::\_\_unicode::\_\_literal\_encoding\_is\_extended\_ascii** ()
- template<typename [\\_CharT](#)>  
constexpr bool **std::\_\_unicode::\_\_literal\_encoding\_is\_unicode** ()
- constexpr bool **std::\_\_unicode::\_\_literal\_encoding\_is\_utf8** ()
- constexpr bool **std::\_\_unicode::\_\_should\_escape\_category** (char32\_t \_\_c) noexcept
- template<typename [\\_CharT](#)>  
constexpr size\_t **std::\_\_unicode::\_\_truncate** ([basic\\_string\\_view](#)< [\\_CharT](#) > &\_\_s, size\_t \_\_max)

## Variables

- constexpr uint32\_t **std::\_\_unicode::\_\_escape\_edges** []
- constexpr uint32\_t **std::\_\_unicode::\_\_gcb\_edges** []
- constexpr int **std::\_\_unicode::\_\_gcb\_shift\_bits**
- constexpr uint32\_t **std::\_\_unicode::\_\_incb\_edges** []
- constexpr char32\_t **std::\_\_unicode::\_\_incb\_linkers** []
- constexpr char32\_t **std::\_\_unicode::\_\_width\_edges** []
- constexpr char32\_t **std::\_\_unicode::\_\_xpicto\_edges** []
- template<typename [\\_Range](#)>  
constexpr bool **std::ranges::enable\_borrowed\_range**< **std::\_\_unicode::\_\_Grapheme\_cluster\_view**< [\\_↔](#)  
**Range** > >
- template<typename [\\_To](#), typename [\\_Range](#)>  
constexpr bool **std::ranges::enable\_borrowed\_range**< **std::\_\_unicode::\_\_Utf\_view**< [\\_To](#), [\\_Range](#) > >

### 6.183.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<format>`.

## 6.184 `uniform_int_dist.h` File Reference

### Classes

- struct [std::uniform\\_int\\_distribution](#)< [\\_IntType](#) >::param\_type
- class [std::uniform\\_int\\_distribution](#)< [\\_IntType](#) >

### Namespaces

- namespace [std](#)

### 6.184.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 6.185 unique\_lock.h File Reference

### Classes

- class [std::unique\\_lock<\\_Mutex>](#)

### Namespaces

- namespace [std](#)

### 6.185.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<mutex>`.

## 6.186 unique\_ptr.h File Reference

### Classes

- struct [std::default\\_delete<\\_Tp>](#)
- struct [std::default\\_delete<\\_Tp\[\]>](#)
- struct [std::hash<unique\\_ptr<\\_Tp, \\_Dp>>](#)
- class [std::unique\\_ptr<\\_Tp, \\_Dp>](#)
- class [std::unique\\_ptr<\\_Tp\[\], \\_Dp>](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- [template<typename \\_Tp, typename \\_Dp, typename \\_Up, typename \\_Ep>  
constexpr bool std::operator< \(const unique\\_ptr<\\_Tp, \\_Dp> &\\_\\_x, const unique\\_ptr<\\_Up, \\_Ep> &\\_\\_y\)](#)
- [template<typename \\_Tp, typename \\_Dp>  
constexpr bool std::operator< \(const unique\\_ptr<\\_Tp, \\_Dp> &\\_\\_x, nullptr\\_t\)](#)
- [template<typename \\_Tp, typename \\_Dp>  
constexpr bool std::operator< \(nullptr\\_t, const unique\\_ptr<\\_Tp, \\_Dp> &\\_\\_x\)](#)
- [template<typename \\_Tp, typename \\_Dp, typename \\_Up, typename \\_Ep>  
constexpr bool std::operator<= \(const unique\\_ptr<\\_Tp, \\_Dp> &\\_\\_x, const unique\\_ptr<\\_Up, \\_Ep> &\\_\\_y\)](#)
- [template<typename \\_Tp, typename \\_Dp>  
constexpr bool std::operator<= \(const unique\\_ptr<\\_Tp, \\_Dp> &\\_\\_x, nullptr\\_t\)](#)
- [template<typename \\_Tp, typename \\_Dp>  
constexpr bool std::operator<= \(nullptr\\_t, const unique\\_ptr<\\_Tp, \\_Dp> &\\_\\_x\)](#)
- [template<typename \\_Tp, typename \\_Dp, typename \\_Up, typename \\_Ep>  
requires three\\_way\\_comparable\\_with<typename unique\\_ptr<\\_Tp, \\_Dp>::pointer, typename unique\\_ptr<\\_Up, \\_Ep>::pointer>  
constexpr compare\\_three\\_way\\_result\\_t<typename unique\\_ptr<\\_Tp, \\_Dp>::pointer, typename unique\\_ptr<\\_Up, \\_Ep>::pointer> std::operator<= \(const unique\\_ptr<\\_Tp, \\_Dp> &\\_\\_x, const unique\\_ptr<\\_Up, \\_Ep> &\\_\\_y\)](#)



- `template<typename _Tp, typename _Dp>`  
`requires three_way_comparable<typename unique\_ptr<_Tp, _Dp>::pointer>`  
`constexpr compare\_three\_way\_result\_t< typename unique\_ptr< _Tp, _Dp >::pointer > std::operator<=>`  
`(const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator== (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator== (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator> (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator> (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator> (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator>= (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator>= (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator>= (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp>`  
`enable\_if<!is\_swappable< _Dp >::value >::type std::swap (unique\_ptr< _Tp, _Dp > &, unique\_ptr< _Tp, _Dp > &)=delete`

### 6.186.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.187 unordered\_map.h File Reference

### Classes

- class `std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >`
- class `std::unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`

### Namespaces

- namespace `std`

### Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>`  
`using std::__umap_hashtable`
- `template<bool _Cache>`  
`using std::\_\_umap\_traits`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>`  
`using std::__ummap_hashtable`
- `template<bool _Cache>`  
`using std::\_\_ummap\_traits`

## Functions

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`bool std::operator==(const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`bool std::operator==(const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`void std::swap(unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`void std::swap(unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::unordered_map(_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::unordered_map(_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
`std::unordered_map(_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Hash = hash< __iter_key_t< _InputIterator > >, typename _Pred = equal_to< __iter_key_t< _InputIterator > >, typename _Allocator = allocator< __iter_val_t< _InputIterator > >, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>`  
`std::unordered_map(_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::unordered_map(initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::unordered_map(initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
`std::unordered_map(initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Hash = hash< _Key >, typename _Pred = equal_to< _Key >, typename _Allocator = allocator< pair< const _Key, _Tp > >, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>`  
`std::unordered_map(initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`

```

std::unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< __iter_key_t<
__iter_val_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to<
__iter_key_t< _InputIterator > >, _Allocator >

• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
std::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator)
-> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t<
_InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >

• template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
std::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >

• template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
std::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >

• template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >

• template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >

• template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >

• template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator >

```

### 6.187.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>`.

## 6.188 unordered\_set.h File Reference

### Classes

- class `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
- class `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`

### Namespaces

- namespace `std`

## Typedefs

- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>  
using std::__umset_hashtable`
- `template<bool _Cache>  
using std::__umset_traits`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>  
using std::__uset_hashtable`
- `template<bool _Cache>  
using std::__uset_traits`

## Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc>  
bool std::operator== (const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>  
bool std::operator== (const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>  
void std::swap (unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>  
void std::swap (unordered\_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered\_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>  
std::unordered_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered\_multiset< typename iterator\_traits< _InputIterator >::value_type, hash< typename iterator\_traits< _InputIterator >::value_type >, equal\_to< typename iterator\_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>  
std::unordered_multiset (_InputIterator, _InputIterator, unordered\_multiset< int >::size_type, _Allocator) -> unordered\_multiset< typename iterator\_traits< _InputIterator >::value_type, hash< typename iterator\_traits< _InputIterator >::value_type >, equal\_to< typename iterator\_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>  
std::unordered_multiset (_InputIterator, _InputIterator, unordered\_multiset< int >::size_type, _Hash, _Allocator) -> unordered\_multiset< typename iterator\_traits< _InputIterator >::value_type, _Hash, equal\_to< typename iterator\_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Hash = hash<typename iterator\_traits<_InputIterator>::value_type>, typename _Pred = equal\_to<typename iterator\_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator\_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>  
std::unordered_multiset (_InputIterator, _InputIterator, unordered\_multiset< int >::size_type={}, _Hash=\_Hash(), _Pred=\_Pred(), _Allocator=\_Allocator()) -> unordered\_multiset< typename iterator\_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >`
- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>  
std::unordered_multiset (initializer\_list< _Tp >, _Allocator) -> unordered\_multiset< _Tp, hash< _Tp >, equal\_to< _Tp >, _Allocator >`

- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_multiset** (`initializer_list`< \_Tp >, `unordered_multiset`< int >::size\_type, \_Allocator) ->  
`unordered_multiset`< \_Tp, `hash`< \_Tp >, `equal_to`< \_Tp >, \_Allocator >
- `template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_multiset** (`initializer_list`< \_Tp >, `unordered_multiset`< int >::size\_type, \_Hash, \_Allocator) ->  
`unordered_multiset`< \_Tp, \_Hash, `equal_to`< \_Tp >, \_Allocator >
- `template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>,`  
`typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_multiset** (`initializer_list`< \_Tp >, `unordered_multiset`< int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> `unordered_multiset`< \_Tp, \_Hash, \_Pred, \_Allocator >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_set** (\_InputIterator, \_InputIterator, \_Allocator) -> `unordered_set`< `typename iterator_traits`< \_InputIterator >::value\_type, `hash`< `typename iterator_traits`< \_InputIterator >::value\_type >, `equal_to`< `typename iterator_traits`< \_InputIterator >::value\_type >, \_Allocator >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_set** (\_InputIterator, \_InputIterator, `unordered_set`< int >::size\_type, \_Allocator) -> `unordered_set`< `typename iterator_traits`< \_InputIterator >::value\_type, `hash`< `typename iterator_traits`< \_InputIterator >::value\_type >, `equal_to`< `typename iterator_traits`< \_InputIterator >::value\_type >, \_Allocator >
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_set** (\_InputIterator, \_InputIterator, `unordered_set`< int >::size\_type, \_Hash, \_Allocator) -> `unordered_set`< `typename iterator_traits`< \_InputIterator >::value\_type, \_Hash, `equal_to`< `typename iterator_traits`< \_InputIterator >::value\_type >, \_Allocator >
- `template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>,`  
`typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_set** (\_InputIterator, \_InputIterator, `unordered_set`< int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> `unordered_set`< `typename iterator_traits`< \_InputIterator >::value\_type, \_Hash, \_Pred, \_Allocator >
- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_set** (`initializer_list`< \_Tp >, \_Allocator) -> `unordered_set`< \_Tp, `hash`< \_Tp >, `equal_to`< \_Tp >, \_Allocator >
- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_set** (`initializer_list`< \_Tp >, `unordered_set`< int >::size\_type, \_Allocator) -> `unordered_set`< \_Tp, `hash`< \_Tp >, `equal_to`< \_Tp >, \_Allocator >
- `template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_set** (`initializer_list`< \_Tp >, `unordered_set`< int >::size\_type, \_Hash, \_Allocator) -> `unordered_set`< \_Tp, \_Hash, `equal_to`< \_Tp >, \_Allocator >
- `template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>,`  
`typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**std::unordered\_set** (`initializer_list`< \_Tp >, `unordered_set`< int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> `unordered_set`< \_Tp, \_Hash, \_Pred, \_Allocator >

### 6.188.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_set>`.

## 6.189 uses\_allocator\_args.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr _Tp std::make_obj_using_allocator (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr _Tp * std::uninitialized_construct_using_allocator (_Tp *__p, const _Alloc &__a, _Args &&... __args)`
- `template<_Std_pair _Tp, typename _Alloc>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, _Up &&, _Vp &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, const pair< _Up, _Vp > &) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, pair< _Up, _Vp > &&) noexcept`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`requires (! _Std_pair<_Tp>)`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &__a, _Args &&... __args) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &__a, piecewise\_construct\_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept`

### 6.189.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.190 utility.h File Reference

### Classes

- struct [std::integer\\_sequence](#)< \_Tp, \_Idx >

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up>>::value>::type, size_t = tuple_size<_Tp>::value>`  
`using std::__enable_if_has_tuple_size`
- `template<size_t __i, typename _Tp>`  
`using std::__tuple_element_t`
- `template<size_t... _Idx>`  
`using std::index\_sequence`
- `template<typename... _Types>`  
`using std::index\_sequence\_for`

- `template<size_t _Num>`  
  using `std::make_index_sequence`
- `template<typename _Tp, _Tp _Num>`  
  using `std::make_integer_sequence`
- `template<size_t __i, typename _Tp>`  
  using `std::tuple_element_t`

## Functions

- `template<typename _Tp, typename... _Types>`  
  constexpr `size_t std::__find_uniq_type_in_pack()`

## Variables

- `template<typename>`  
  constexpr `bool std::__is_in_place_index_v`
- `template<size_t _Nm>`  
  constexpr `bool std::__is_in_place_index_v<in_place_index_t<_Nm>>`
- `template<typename>`  
  constexpr `bool std::__is_in_place_type_v`
- `template<typename _Tp>`  
  constexpr `bool std::__is_in_place_type_v<in_place_type_t<_Tp>>`
- `template<typename _Range>`  
  constexpr `bool std::ranges::__detail::__is_subrange`
- constexpr `_Swallow_assign std::ignore`
- constexpr `in_place_t std::in_place`
- `template<size_t _Idx>`  
  constexpr `in_place_index_t<_Idx> std::in_place_index`
- `template<typename _Tp>`  
  constexpr `in_place_type_t<_Tp> std::in_place_type`
- `template<typename _Tp>`  
  constexpr `size_t std::tuple_size_v`

### 6.190.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

This file contains the parts of `<utility>` needed by other headers, so they don't need to include the whole of `<utility>`.

## 6.191 valarray\_after.h File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Macros

- `#define _DEFINE_EXPR_BINARY_FUNCTION(_Fun, _UFun)`
- `#define _DEFINE_EXPR_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_EXPR_UNARY_FUNCTION(_Name, _UName)`
- `#define _DEFINE_EXPR_UNARY_OPERATOR(_Op, _Name)`

## Functions

- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Abs, _Expr, _Dom >, typename _Dom::value_type > std::abs (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Abs, _ValArray, _Tp >, _Tp > std::abs (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Acos, _Expr, _Dom >, typename _Dom::value_type > std::acos (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Acos, _ValArray, _Tp >, _Tp > std::acos (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Asin, _Expr, _Dom >, typename _Dom::value_type > std::asin (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > std::asin (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > std::atan (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > std::atan (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename ↵`  
`_Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename`  
`_Dom::value_type &__t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename ↵`  
`_Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`  
`typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::↵`  
`atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename ↵`  
`_Dom2::value_type > &__e2)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ↵`  
`_Dom::value_type > std::atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename ↵`  
`_Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const typename`  
`valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::atan2 (const valarray<`  
`_Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename ↵`  
`_Dom::value_type > std::atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Cos, _Expr, _Dom >, typename _Dom::value_type > std::cos (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`



- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Cosh, _Expr, _Dom >, typename _Dom::value_type > std::cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > std::exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Log, _Expr, _Dom >, typename _Dom::value_type > std::log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Log, _ValArray, _Tp >, _Tp > std::log (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Log10, _Expr, _Dom >, typename _Dom::value_type > std::log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Log10, _ValArray, _Tp >, _Tp > std::log10 (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_not_equal_to, typename _Dom1::value_type >::result_type > std::operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::_modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::_modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_modulus, typename _Dom1::value_type >::result_type > std::operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > std::operator% (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > std::operator% (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`_Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`bitwise_and, typename _Dom1::value_type >::result_type > std::operator& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const`  
`_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`logical_and, typename _Dom1::value_type >::result_type > std::operator&& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__multiplies, typename _Dom1::value_type >::result_type > std::operator* (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __`  
`__fun< struct std::__plus, typename _Dom::value_type >::result_type > std::operator+ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __`  
`__fun< struct std::__plus, typename _Dom::value_type >::result_type > std::operator+ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__plus,`  
`typename _Dom1::value_type >::result_type > std::operator+ (const _Expr< _Dom1, typename _Dom1::`  
`value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __`  
`fun< struct std::__plus, typename _Dom::value_type >::result_type > std::operator+ (const typename _Dom::`  
`value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __`  
`__fun< struct std::__plus, typename _Dom::value_type >::result_type > std::operator+ (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __`  
`__fun< struct std::__minus, typename _Dom::value_type >::result_type > std::operator- (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __`  
`__fun< struct std::__minus, typename _Dom::value_type >::result_type > std::operator- (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__minus, typename _Dom1::value_type >::result_type > std::operator- (const _Expr< _Dom1, typename __`  
`Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__minus, typename _Dom::value_type >::result_type > std::operator- (const typename __`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type>::__minus, typename _Dom::value_type>::result_type> std::operator- (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::__divides, typename _Dom::value_type>::result_type> std::operator/ (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::__divides, typename _Dom::value_type>::result_type> std::operator/ (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2>, typename _Dom1::value_type>::__divides, typename _Dom1::value_type>::result_type> std::operator/ (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type>::__divides, typename _Dom::value_type>::result_type> std::operator/ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type>::__divides, typename _Dom::value_type>::result_type> std::operator/ (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::__less, typename _Dom::value_type>::result_type> std::operator< (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::__less, typename _Dom::value_type>::result_type> std::operator< (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2>, typename _Dom1::value_type>::__less, typename _Dom1::value_type>::result_type> std::operator< (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type>::__less, typename _Dom::value_type>::result_type> std::operator< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type>::__less, typename _Dom::value_type>::result_type> std::operator< (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::__shift_left, typename _Dom::value_type>::result_type> std::operator<< (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::__shift_left, typename _Dom::value_type>::result_type> std::operator<< (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`

Generated by Doxygen

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _`  
`_fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _`  
`Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`greater, typename _Dom1::value_type >::result_type > std::operator> (const _Expr< _Dom1, typename _`  
`Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name _fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type >`  
`&__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >,`   
`typename _fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`greater_equal, typename _Dom1::value_type >::result_type > std::operator>= (const _Expr< _Dom1, type-`  
`name _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >,`   
`typename _fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name _fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`shift_right, typename _Dom1::value_type >::result_type > std::operator>> (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`



- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const type-`  
`name _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const`  
`valarray< typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const _Expr<`  
`_Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const`  
`_Expr< _Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__bitwise_xor, typename _Dom1::value_type >::result_type > std::operator^ (const _Expr< _Dom1, typename`  
`_Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const`  
`typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const valarray<`  
`typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr<`  
`_Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr<`  
`_Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__bitwise_or, typename _Dom1::value_type >::result_type > std::operator| (const _Expr< _Dom1, typename`  
`_Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const typename`  
`_Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const valarray<`  
`typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr<`  
`_Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::_logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::↵`  
`_logical_or, typename _Dom1::value_type >::result_type > std::operator|| (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::_logical_or, typename _Dom::value_type >::result_type > std::operator|| (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::_logical_or, typename _Dom::value_type >::result_type > std::operator|| (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename ↵`  
`_Dom::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename`  
`_Dom::value_type &__t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom↵`  
`::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename`  
`_Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std↵`  
`::pow (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename ↵`  
`_Dom2::value_type > &__e2)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ↵`  
`_Dom::value_type > std::pow (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom↵`  
`::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::pow (const typename`  
`valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp`  
`> &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom↵`  
`::value_type > std::pow (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename`  
`_Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Sin, _Expr, _Dom >, typename _Dom::value_type > std::sin (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Sin, _ValArray, _Tp >, _Tp > std::sin (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Sinh, _Expr, _Dom >, typename _Dom::value_type > std::sinh (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`



- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Sqrt, _Expr, _Dom >, typename _Dom::value_type > std::sqrt (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Tan, _Expr, _Dom >, typename _Dom::value_type > std::tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Tan, _ValArray, _Tp >, _Tp > std::tan (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Tanh, _Expr, _Dom >, typename _Dom::value_type > std::tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray< _Tp > &__v)`

### 6.191.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.192 `valarray_array.h` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

### Functions

- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`

- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp>`  
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp>`  
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp & __t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp & __t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp & __t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp & __t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp & __t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp & __t)`
- `template<typename _Tp>`  
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `template<typename _Tp>`  
`_Tp * std::__valarray_get_storage (size_t)`
- `template<typename _Ta>`  
`_Ta::value_type std::__valarray_max (const _Ta & __a)`
- `template<typename _Ta>`  
`_Ta::value_type std::__valarray_min (const _Ta & __a)`
- `void std::__valarray_release_memory (void * __p)`
- `template<typename _Tp>`  
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`

- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, const Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented_bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented_bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented_bitwise_or (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented_bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`  
`> __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`size_t > __i)`

- `template<typename _Tp>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`

### 6.192.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.



## 6.193 valarray\_array.tcc File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _VALARRAY_ARRAY_TCC`

### Functions

- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`

#### 6.193.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.194 valarray\_before.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

#### 6.194.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.195 vector.tcc File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _VECTOR_TCC`

#### 6.195.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 6.196 version.h File Reference

### Macros

- `#define __glibcxx_addressof_constexpr`
- `#define __glibcxx_allocator_traits_is_always_equal`
- `#define __glibcxx_any`
- `#define __glibcxx_apply`
- `#define __glibcxx_array_constexpr`
- `#define __glibcxx_as_const`
- `#define __glibcxx_assume_aligned`
- `#define __glibcxx_atomic_flag_test`
- `#define __glibcxx_atomic_float`
- `#define __glibcxx_atomic_is_always_lock_free`
- `#define __glibcxx_atomic_ref`
- `#define __glibcxx_atomic_shared_ptr`
- `#define __glibcxx_atomic_value_initialization`
- `#define __glibcxx_atomic_wait`
- `#define __glibcxx_barrier`
- `#define __glibcxx_bind_front`
- `#define __glibcxx_bit_cast`
- `#define __glibcxx_bitops`
- `#define __glibcxx_bool_constant`
- `#define __glibcxx_bounded_array_traits`
- `#define __glibcxx_boyer_moore_searcher`
- `#define __glibcxx_byte`
- `#define __glibcxx_chrono`
- `#define __glibcxx_chrono_cxx20`
- `#define __glibcxx_chrono_udls`
- `#define __glibcxx_clamp`
- `#define __glibcxx_complex_udls`
- `#define __glibcxx_concepts`
- `#define __glibcxx_constexpr_algorithms`
- `#define __glibcxx_constexpr_char_traits`
- `#define __glibcxx_constexpr_complex`
- `#define __glibcxx_constexpr_dynamic_alloc`
- `#define __glibcxx_constexpr_functional`
- `#define __glibcxx_constexpr_iterator`

- `#define __glibcxx_constexpr_memory`
- `#define __glibcxx_constexpr_numeric`
- `#define __glibcxx_constexpr_string`
- `#define __glibcxx_constexpr_string_view`
- `#define __glibcxx_constexpr_tuple`
- `#define __glibcxx_constexpr_utility`
- `#define __glibcxx_constexpr_vector`
- `#define __glibcxx_constrained_equality`
- `#define __glibcxx_coroutine`
- `#define __glibcxx_enable_shared_from_this`
- `#define __glibcxx_endian`
- `#define __glibcxx_erase_if`
- `#define __glibcxx_exchange_function`
- `#define __glibcxx_execution`
- `#define __glibcxx_filesystem`
- `#define __glibcxx_format`
- `#define __glibcxx_format_uchar`
- `#define __glibcxx_gcd`
- `#define __glibcxx_gcd_lcm`
- `#define __glibcxx_generic_associative_lookup`
- `#define __glibcxx_generic_unordered_lookup`
- `#define __glibcxx_has_unique_object_representations`
- `#define __glibcxx_hypot`
- `#define __glibcxx_incomplete_container_elements`
- `#define __glibcxx_int_pow2`
- `#define __glibcxx_integer_comparison_functions`
- `#define __glibcxx_integer_sequence`
- `#define __glibcxx_integral_constant_callable`
- `#define __glibcxx_interpolate`
- `#define __glibcxx_invoke`
- `#define __glibcxx_is_aggregate`
- `#define __glibcxx_is_constant_evaluated`
- `#define __glibcxx_is_final`
- `#define __glibcxx_is_invocable`
- `#define __glibcxx_is_layout_compatible`
- `#define __glibcxx_is_nothrow_convertible`
- `#define __glibcxx_is_null_pointer`
- `#define __glibcxx_is_pointer_interconvertible`
- `#define __glibcxx_is_swappable`
- `#define __glibcxx_jthread`
- `#define __glibcxx_latch`
- `#define __glibcxx_laundry`
- `#define __glibcxx_lcm`
- `#define __glibcxx_list_remove_return_type`
- `#define __glibcxx_logical_traits`
- `#define __glibcxx_make_from_tuple`
- `#define __glibcxx_make_obj_using_allocator`
- `#define __glibcxx_make_reverse_iterator`
- `#define __glibcxx_make_unique`
- `#define __glibcxx_map_try_emplace`
- `#define __glibcxx_math_constants`

- `#define __glibcxx_math_spec_funcs`
- `#define __glibcxx_math_special_functions`
- `#define __glibcxx_memory_resource`
- `#define __glibcxx_move_iterator_concept`
- `#define __glibcxx_node_extract`
- `#define __glibcxx_nonmember_container_access`
- `#define __glibcxx_not_fn`
- `#define __glibcxx_null_iterators`
- `#define __glibcxx_optional`
- `#define __glibcxx_parallel_algorithm`
- `#define __glibcxx_polymorphic_allocator`
- `#define __glibcxx_quoted_string_io`
- `#define __glibcxx_ranges`
- `#define __glibcxx_raw_memory_algorithms`
- `#define __glibcxx_remove_cvref`
- `#define __glibcxx_result_of_sfinae`
- `#define __glibcxx_robust_nonmodifying_seq_ops`
- `#define __glibcxx_sample`
- `#define __glibcxx_scoped_lock`
- `#define __glibcxx_semaphore`
- `#define __glibcxx_shared_mutex`
- `#define __glibcxx_shared_ptr_arrays`
- `#define __glibcxx_shared_ptr_weak_type`
- `#define __glibcxx_shared_timed_mutex`
- `#define __glibcxx_shift`
- `#define __glibcxx_smart_ptr_for_overwrite`
- `#define __glibcxx_source_location`
- `#define __glibcxx_span`
- `#define __glibcxx_ssize`
- `#define __glibcxx_starts_ends_with`
- `#define __glibcxx_string_udls`
- `#define __glibcxx_string_view`
- `#define __glibcxx_three_way_comparison`
- `#define __glibcxx_to_address`
- `#define __glibcxx_transformation_trait_aliases`
- `#define __glibcxx_transparent_operators`
- `#define __glibcxx_tuple_element_t`
- `#define __glibcxx_tuples_by_type`
- `#define __glibcxx_type_identity`
- `#define __glibcxx_type_trait_variable_templates`
- `#define __glibcxx_uncaught_exceptions`
- `#define __glibcxx_unordered_map_try_emplace`
- `#define __glibcxx_unwrap_ref`
- `#define __glibcxx_variant`
- `#define __glibcxx_void_t`

### 6.196.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<version>`.

## 6.197 bitset File Reference

### Classes

- struct [std::\\_Base\\_bitset<\\_Nw>](#)
- struct [std::\\_Base\\_bitset<0>](#)
- struct [std::\\_Base\\_bitset<1>](#)
- class [std::bitset<\\_Nb>](#)
- struct [std::hash<::bitset<\\_Nb>>](#)
- class [std::bitset<\\_Nb>::reference](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_bitset`
- `#define __glibcxx_want_constexpr_bitset`
- `#define _GLIBCXX_BITSET`
- `#define _GLIBCXX_BITSET_BITS_PER_ULL`
- `#define _GLIBCXX_BITSET_BITS_PER_WORD`
- `#define _GLIBCXX_BITSET_WORDS(__n)`

### Typedefs

- `template<typename _CharT>`  
`using std::\_bitset::\_\_string`

### Functions

- `template<size_t _Nb>`  
`constexpr bitset<\_Nb> std::operator& (const bitset<\_Nb> &__x, const bitset<\_Nb> &__y) noexcept`
- `template<size_t _Nb>`  
`constexpr bitset<\_Nb> std::operator^ (const bitset<\_Nb> &__x, const bitset<\_Nb> &__y) noexcept`
- `template<size_t _Nb>`  
`constexpr bitset<\_Nb> std::operator| (const bitset<\_Nb> &__x, const bitset<\_Nb> &__y) noexcept`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_ostream<\_CharT, \_Traits> & std::operator<< (std::basic\_ostream<\_CharT, \_Traits> &__os, const bitset<\_Nb> &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_istream<\_CharT, \_Traits> & std::operator>> (std::basic\_istream<\_CharT, \_Traits> &__is, bitset<\_Nb> &__x)`

#### 6.197.1 Detailed Description

This is a Standard C++ Library header.

## 6.198 debug/bitset File Reference

### Classes

- class [std::\\_\\_debug::bitset<\\_Nb>](#)
- struct [std::hash<\\_\\_debug::bitset<\\_Nb>>](#)

## Namespaces

- namespace `std`
- namespace `std::__debug`

## Functions

- `template<size_t _Nb>`  
`constexpr bitset< _Nb > std::__debug::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`  
`noexcept`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > & std::__debug::operator<< (std::basic_ostream< _CharT, _Traits >`  
`&__os, const bitset< _Nb > &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & std::__debug::operator>> (std::basic_istream< _CharT, _Traits >`  
`&__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > std::__debug::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`  
`noexcept`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > std::__debug::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`  
`noexcept`

### 6.198.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.199 cassert File Reference

### 6.199.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.200 ccomplex File Reference

### Macros

- `#define _GLIBCXX_CCOMPLEX`

### 6.200.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

## 6.201 tr1/complex File Reference

### Macros

- `#define _GLIBCXX_TR1_CCOMPLEX`

### 6.201.1 Detailed Description

This is a TR1 C++ Library header.

## 6.202 cctype File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_CCTYPE`

### 6.202.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `ctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.203 tr1/cctype File Reference

### Macros

- `#define _GLIBCXX_TR1_CCTYPE`

### 6.203.1 Detailed Description

This is a TR1 C++ Library header.

## 6.204 cerrno File Reference

### Macros

- `#define _GLIBCXX_CERRNO`
- `#define errno`

### 6.204.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.205 cenv File Reference

### Macros

- `#define _GLIBCXX_CFENV`

### 6.205.1 Detailed Description

This is a Standard C++ Library header.

## 6.206 tr1/cfenv File Reference

### Macros

- `#define _GLIBCXX_TR1_CFENV`

#### 6.206.1 Detailed Description

This is a TR1 C++ Library header.

## 6.207 cfloat File Reference

### Macros

- `#define _GLIBCXX_CFLOAT`
- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

#### 6.207.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.208 tr1/cfloat File Reference

### Macros

- `#define _GLIBCXX_TR1_CFLOAT`

#### 6.208.1 Detailed Description

This is a TR1 C++ Library header.

## 6.209 charconv File Reference

### Classes

- struct [std::from\\_chars\\_result](#)
- struct [std::to\\_chars\\_result](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Macros

- `#define __glibcxx_want_constexpr_charconv`
- `#define __glibcxx_want_to_chars`
- `#define _GLIBCXX_CHARCONV`
- `#define _GLIBCXX_TO_CHARS(T)`



## Typedefs

- `template<typename _Tp>`  
using `std::__detail::__unsigned_least_t`

## Enumerations

- enum class `std::chars_format` { `scientific` , `fixed` , `hex` , `general` }

## Functions

- `template<bool _DecOnly, typename _Tp>`  
constexpr bool `std::__detail::__from_chars_alnum` (const char \*\_\_first, const char \*\_\_last, \_Tp &\_\_val, int \_\_base)
- `template<bool _DecOnly = false>`  
constexpr unsigned char `std::__detail::__from_chars_alnum_to_val` (unsigned char \_\_c)
- `from_chars_result std::__from_chars_bfloat16_t` (const char \*\_\_first, const char \*\_\_last, float &\_\_value, `chars_format` \_\_fmt=`chars_format::general`) noexcept
- `from_chars_result std::__from_chars_float16_t` (const char \*\_\_first, const char \*\_\_last, float &\_\_value, `chars_format` \_\_fmt=`chars_format::general`) noexcept
- `template<bool _DecOnly, typename _Tp>`  
constexpr bool `std::__detail::__from_chars_pow2_base` (const char \*\_\_first, const char \*\_\_last, \_Tp &\_\_val, int \_\_base)
- `template<typename _Tp>`  
constexpr bool `std::__detail::__raise_and_add` (\_Tp &\_\_val, int \_\_base, unsigned char \_\_c)
- `template<typename _Tp>`  
constexpr `to_chars_result std::__detail::__to_chars` (char \*\_\_first, char \*\_\_last, \_Tp \_\_val, int \_\_base) noexcept
- `template<typename _Tp>`  
constexpr `to_chars_result std::__detail::__to_chars_10` (char \*\_\_first, char \*\_\_last, \_Tp \_\_val) noexcept
- `template<typename _Tp>`  
constexpr `to_chars_result std::__detail::__to_chars_16` (char \*\_\_first, char \*\_\_last, \_Tp \_\_val) noexcept
- `template<typename _Tp>`  
constexpr `to_chars_result std::__detail::__to_chars_2` (char \*\_\_first, char \*\_\_last, \_Tp \_\_val) noexcept
- `template<typename _Tp>`  
constexpr `to_chars_result std::__detail::__to_chars_8` (char \*\_\_first, char \*\_\_last, \_Tp \_\_val) noexcept
- `template<typename _Tp>`  
constexpr `to_chars_result std::__to_chars_i` (char \*\_\_first, char \*\_\_last, \_Tp \_\_value, int \_\_base=10)
- `template<typename _Tp>`  
constexpr unsigned `std::__detail::__to_chars_len` (\_Tp \_\_value, int \_\_base) noexcept
- `template<typename _Tp>`  
constexpr unsigned `std::__detail::__to_chars_len_2` (\_Tp \_\_value) noexcept
- `template<typename _Tp, enable_if_t<__or_<__is_standard_integer<_Tp>, is_same<char, remove_cv_t<_Tp>>>::value, int> = 0>`  
constexpr `from_chars_result std::from_chars` (const char \*\_\_first, const char \*\_\_last, \_Tp &\_\_value, int \_\_base=10)
- `from_chars_result std::from_chars` (const char \*\_\_first, const char \*\_\_last, double &\_\_value, `chars_format` \_\_fmt=`chars_format::general`) noexcept
- `from_chars_result std::from_chars` (const char \*\_\_first, const char \*\_\_last, float &\_\_value, `chars_format` \_\_fmt=`chars_format::general`) noexcept
- `from_chars_result std::from_chars` (const char \*\_\_first, const char \*\_\_last, long double &\_\_value, `chars_format` \_\_fmt=`chars_format::general`) noexcept
- constexpr `chars_format std::operator&` (`chars_format` \_\_lhs, `chars_format` \_\_rhs) noexcept
- constexpr `chars_format & std::operator&=` (`chars_format` &\_\_lhs, `chars_format` \_\_rhs) noexcept

- constexpr [chars\\_format](#) [std::operator^](#) ([chars\\_format](#) \_\_lhs, [chars\\_format](#) \_\_rhs) noexcept
- constexpr [chars\\_format](#) & [std::operator^](#) = ([chars\\_format](#) & \_\_lhs, [chars\\_format](#) \_\_rhs) noexcept
- constexpr [chars\\_format](#) [std::operator|](#) ([chars\\_format](#) \_\_lhs, [chars\\_format](#) \_\_rhs) noexcept
- constexpr [chars\\_format](#) & [std::operator|](#) = ([chars\\_format](#) & \_\_lhs, [chars\\_format](#) \_\_rhs) noexcept
- constexpr [chars\\_format](#) [std::operator~](#) ([chars\\_format](#) \_\_fmt) noexcept
- [to\\_chars\\_result](#) [std::to\\_chars](#) (char \*, char \*, bool, int=10)=delete
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, char \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, signed char \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, signed int \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, signed long \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, signed long long \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, signed short \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, unsigned char \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, unsigned int \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, unsigned long \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, unsigned long long \_\_value, int \_\_base=10)
- constexpr [to\\_chars\\_result](#) [std::to\\_chars](#) (char \* \_\_first, char \* \_\_last, unsigned short \_\_value, int \_\_base=10)

### 6.209.1 Detailed Description

This is a Standard C++ Library header.

## 6.210 chrono File Reference

### Classes

- class [std::chrono::tzdb\\_list::const\\_iterator](#)
- class [std::chrono::gps\\_clock](#)
- class [std::chrono::hh\\_mm\\_ss< \\_Duration >](#)
- class [std::chrono::tai\\_clock](#)
- class [std::chrono::tzdb\\_list](#)
- class [std::chrono::utc\\_clock](#)

### Namespaces

- namespace [std](#)
- namespace [std::chrono](#)
- namespace [std::literals](#)
- namespace [std::literals::chrono\\_literals](#)

### Macros

- [#define](#) [\\_\\_glibcxx\\_want\\_chrono](#)
- [#define](#) [\\_\\_glibcxx\\_want\\_chrono\\_udls](#)
- [#define](#) [\\_GLIBCXX\\_CHRONO](#)

## Typedefs

- using **std::chrono::\_\_detail::\_\_months\_years\_conversion\_disambiguator**
- using **std::chrono::gps\_seconds**
- template<typename \_Duration>  
using **std::chrono::gps\_time**
- using **std::chrono::local\_days**
- using **std::chrono::local\_seconds**
- template<typename \_Duration>  
using **std::chrono::local\_time**
- using **std::chrono::tai\_seconds**
- template<typename \_Duration>  
using **std::chrono::tai\_time**
- using **std::chrono::utc\_seconds**
- template<typename \_Duration>  
using **std::chrono::utc\_time**
- using **std::chrono::zoned\_seconds**

## Enumerations

- enum class **choose** { **earliest** , **latest** }

## Functions

- template<unsigned \_\_d, typename \_Tp>  
constexpr unsigned **std::chrono::\_\_detail::\_\_add\_modulo** (unsigned \_\_x, \_Tp \_\_y)
- leap\_second\_info **std::chrono::\_\_detail::\_\_get\_leap\_second\_info** (sys\_seconds \_\_ss, bool \_\_is\_utc)
- template<unsigned \_\_d, typename \_Tp>  
constexpr auto **std::chrono::\_\_detail::\_\_modulo\_offset** ()
- template<unsigned \_\_d, typename \_Tp>  
constexpr unsigned **std::chrono::\_\_detail::\_\_sub\_modulo** (unsigned \_\_x, \_Tp \_\_y)
- template<typename \_Duration>  
void **std::chrono::\_\_throw\_bad\_local\_time** (const local\_time< \_Duration > &\_\_tp, const local\_info &\_\_i)
- template<typename \_DestClock, typename \_SourceClock, typename \_Duration>  
requires \_\_detail::\_\_clock\_convs<\_DestClock, \_SourceClock, \_Duration> || \_\_detail::\_\_clock\_convs\_sys<\_DestClock, \_SourceClock, \_Duration> || \_\_detail::\_\_clock\_convs\_utc<\_DestClock, \_SourceClock, \_Duration> || \_\_detail::\_\_clock\_convs\_sys\_utc<\_DestClock, \_SourceClock, \_Duration> || \_\_detail::\_\_clock\_convs\_utc\_sys<\_DestClock, \_SourceClock, \_Duration>  
auto **std::chrono::clock\_cast** (const time\_point< \_SourceClock, \_Duration > &\_\_t)
- const time\_zone \* **std::chrono::current\_zone** ()
- template<typename \_Duration>  
leap\_second\_info **std::chrono::get\_leap\_second\_info** (const utc\_time< \_Duration > &\_\_ut)
- const tzdb & **std::chrono::get\_tzdb** ()
- tzdb\_list & **std::chrono::get\_tzdb\_list** ()
- constexpr bool **std::chrono::is\_am** (const hours &\_\_h) noexcept
- constexpr bool **std::chrono::is\_pm** (const hours &\_\_h) noexcept
- const time\_zone \* **std::chrono::locate\_zone** (string\_view \_\_tz\_name)
- constexpr hours **std::chrono::make12** (const hours &\_\_h) noexcept
- constexpr hours **std::chrono::make24** (const hours &\_\_h, bool \_\_is\_pm) noexcept
- constexpr chrono::day **std::literals::chrono\_literals::operator""d** (unsigned long long \_\_d) noexcept
- constexpr chrono::year **std::literals::chrono\_literals::operator""y** (unsigned long long \_\_y) noexcept
- template<typename \_Dur1, typename \_TZPtr1, typename \_Dur2, typename \_TZPtr2>  
bool **std::chrono::operator==** (const zoned\_time< \_Dur1, \_TZPtr1 > &\_\_x, const zoned\_time< \_Dur2, \_TZPtr2 > &\_\_y)

- `const tzdb & std::chrono::reload_tzdb ()`
- `string std::chrono::remote_version ()`
- `std::chrono::zoned_time () -> zoned_time< seconds >`
- `template<typename _TimeZonePtrOrName>  
std::chrono::zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation< _TimeZonePtrOrName > >`
- `template<typename _TimeZonePtrOrName, typename _Duration>  
std::chrono::zoned_time (_TimeZonePtrOrName &&, local_time< _Duration >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >`
- `template<typename _TimeZonePtrOrName, typename _Duration>  
std::chrono::zoned_time (_TimeZonePtrOrName &&, sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >`
- `template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>  
std::chrono::zoned_time (_TimeZonePtrOrName &&, zoned_time< _Duration, _TimeZonePtr2 >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >`
- `template<typename _Duration>  
std::chrono::zoned_time (sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >`

## Variables

- `constexpr unsigned std::chrono::__detail::__days_per_month [12]`
- `template<typename _Tp, typename _Clock>  
constexpr bool std::chrono::__is_time_point_for_v`
- `template<typename _Clock, typename _Duration>  
constexpr bool std::chrono::__is_time_point_for_v< time_point< _Clock, _Duration >, _Clock >`
- `constexpr month std::chrono::April`
- `constexpr month std::chrono::August`
- `constexpr month std::chrono::December`
- `constexpr month std::chrono::February`
- `constexpr weekday std::chrono::Friday`
- `template<> constexpr bool std::chrono::is_clock_v< gps_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< tai_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< utc_clock >`
- `constexpr month std::chrono::January`
- `constexpr month std::chrono::July`
- `constexpr month std::chrono::June`
- `constexpr last_spec std::chrono::last`
- `constexpr month std::chrono::March`
- `constexpr month std::chrono::May`
- `constexpr weekday std::chrono::Monday`
- `constexpr month std::chrono::November`
- `constexpr month std::chrono::October`
- `constexpr weekday std::chrono::Saturday`
- `constexpr month std::chrono::September`
- `constexpr weekday std::chrono::Sunday`
- `constexpr weekday std::chrono::Thursday`
- `constexpr weekday std::chrono::Tuesday`
- `constexpr weekday std::chrono::Wednesday`

### 6.210.1 Detailed Description

This is a Standard C++ Library header.

## 6.211 experimental/chrono File Reference

### Namespaces

- namespace [std](#)
- namespace [std::chrono](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_CHRONO`

### Variables

- `template<typename _Rep>`  
`constexpr bool std::chrono::experimental::treat\_as\_floating\_point\_v`

### 6.211.1 Detailed Description

This is a TS C++ Library header.

## 6.212 cinttypes File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CINTTYPES`

### 6.212.1 Detailed Description

This is a Standard C++ Library header.

## 6.213 tr1/cinttypes File Reference

### Macros

- `#define _GLIBCXX_TR1_CINTTYPES`

### 6.213.1 Detailed Description

This is a TR1 C++ Library header.

## 6.214 ciso646 File Reference

### 6.214.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, which is empty in C++.

Since

C++11 (removed in C++20)

## 6.215 climits File Reference

### Macros

- `#define _GLIBCXX_CLIMITS`
- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

#### 6.215.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.216 tr1/climits File Reference

### Macros

- `#define _GLIBCXX_TR1_CLIMITS`

#### 6.216.1 Detailed Description

This is a TR1 C++ Library header.

## 6.217 clocale File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CLOCALE`

#### 6.217.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.218 cmath File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define __glibcxx_want_hypot`
- `#define __glibcxx_want_interpolate`
- `#define _GLIBCXX_CMATH`
- `#define _GLIBCXX_INCLUDE_NEXT_C_HEADERS`

## Functions

- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::acos ( _Tp __x)`
- `constexpr float std::acos (float __x)`
- `constexpr long double std::acos (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::acos(declval<double>()))), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::acos (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::acosh ( _Tp __x)`
- `constexpr float std::acosh (float __x)`
- `constexpr long double std::acosh (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::acosh(declval<double>()))), <_Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::acosh (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::asin ( _Tp __x)`
- `constexpr float std::asin (float __x)`
- `constexpr long double std::asin (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::asin(declval<double>()))), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::asin (simd< _Tp, <_Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::asinh ( _Tp __x)`
- `constexpr float std::asinh (float __x)`
- `constexpr long double std::asinh (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::asinh(declval<double>()))), <_Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::asinh (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::atan ( _Tp __x)`
- `constexpr float std::atan (float __x)`
- `constexpr long double std::atan (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::atan(declval<double>()))), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::atan (simd< _Tp, <_Abi > __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type std::atan2 ( _Tp __y, _Up __x)`

- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::atan2(declval<double>()), _Arg2::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::atan2 (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)`
- `constexpr float std::atan2 (float __y, float __x)`
- `constexpr long double std::atan2 (long double __y, long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::atanh (_Tp __x)`
- `constexpr float std::atanh (float __x)`
- `constexpr long double std::atanh (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::atanh(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::atanh (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::cbrt (_Tp __x)`
- `constexpr float std::cbrt (float __x)`
- `constexpr long double std::cbrt (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::cbrt(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::cbrt (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::ceil (_Tp __x)`
- `constexpr float std::ceil (float __x)`
- `constexpr long double std::ceil (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::ceil(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::ceil (simd< _Tp, _Abi > __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type std::copysign (_Tp __x, _Up __y)`
- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>`  
`enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > std::copysign (const simd< _Tp, _Abi > &__x, const simd< _Tp, _Abi > &__y)`
- `constexpr float std::copysign (float __x, float __y)`
- `constexpr long double std::copysign (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::cos (_Tp __x)`
- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>`  
`enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > std::cos (const simd< _Tp, _Abi > &__x)`
- `constexpr float std::cos (float __x)`
- `constexpr long double std::cos (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::cosh (_Tp __x)`
- `constexpr float std::cosh (float __x)`
- `constexpr long double std::cosh (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::cosh(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::cosh (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::erf (_Tp __x)`



- constexpr float **std::erf** (float \_\_x)
- constexpr long double **std::erf** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::erf(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::erf** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::erfc** (\_Tp \_\_x)
- constexpr float **std::erfc** (float \_\_x)
- constexpr long double **std::erfc** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::erfc(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::erfc** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::exp** (\_Tp \_\_x)
- constexpr float **std::exp** (float \_\_x)
- constexpr long double **std::exp** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::exp(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::exp** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::exp2** (\_Tp \_\_x)
- constexpr float **std::exp2** (float \_\_x)
- constexpr long double **std::exp2** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::exp2(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::exp2** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::expm1** (\_Tp \_\_x)
- constexpr float **std::expm1** (float \_\_x)
- constexpr long double **std::expm1** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::expm1(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::expm1** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::fabs** (\_Tp \_\_x)
- constexpr float **std::fabs** (float \_\_x)
- constexpr long double **std::fabs** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::fabs(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::fabs** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fdim** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp, typename \_Abi, typename..., typename \_Arg2 = \_Extra\_argument\_type< \_Tp, \_Tp, \_Abi>, typename \_R = \_Math\_return\_type\_t< decltype(std::fdim(declval<double>(), \_Arg2::declval())), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::fdim** (const simd< \_Tp, \_Abi > &\_\_x, const typename \_Arg2::type &\_\_y)

- constexpr float **std::fdim** (float \_\_x, float \_\_y)
- constexpr long double **std::fdim** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::floor** (\_Tp \_\_x)
- constexpr float **std::floor** (float \_\_x)
- constexpr long double **std::floor** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::floor(declval<double>())), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::floor** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp, typename \_Up, typename \_Vp>  
constexpr \_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Up, \_Vp >::\_\_type **std::fma** (\_Tp \_\_x, \_Up \_\_y, \_Vp \_\_z)
- template<typename \_Tp, typename \_Abi, typename..., typename \_Arg2 = \_Extra\_argument\_type< \_Tp, \_Tp, \_Abi>, typename \_Arg3 = \_Extra\_argument\_type< \_Tp, \_Tp, \_Abi>, typename \_R = \_Math\_return\_type\_t< decltype(std::fma(declval<double>(), \_Arg2::declval(), \_Arg3::declval())), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::fma** (const simd< \_Tp, \_Abi > &\_\_x, const typename \_Arg2::type &\_\_y, const typename \_Arg3::type &\_\_z)
- constexpr float **std::fma** (float \_\_x, float \_\_y, float \_\_z)
- constexpr long double **std::fma** (long double \_\_x, long double \_\_y, long double \_\_z)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fmax** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp, typename \_Abi, typename..., typename \_Arg2 = \_Extra\_argument\_type< \_Tp, \_Tp, \_Abi>, typename \_R = \_Math\_return\_type\_t< decltype(std::fmax(declval<double>(), \_Arg2::declval())), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::fmax** (const simd< \_Tp, \_Abi > &\_\_x, const typename \_Arg2::type &\_\_y)
- constexpr float **std::fmax** (float \_\_x, float \_\_y)
- constexpr long double **std::fmax** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fmin** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp, typename \_Abi, typename..., typename \_Arg2 = \_Extra\_argument\_type< \_Tp, \_Tp, \_Abi>, typename \_R = \_Math\_return\_type\_t< decltype(std::fmin(declval<double>(), \_Arg2::declval())), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::fmin** (const simd< \_Tp, \_Abi > &\_\_x, const typename \_Arg2::type &\_\_y)
- constexpr float **std::fmin** (float \_\_x, float \_\_y)
- constexpr long double **std::fmin** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fmod** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp, typename \_Abi, typename..., typename \_Arg2 = \_Extra\_argument\_type< \_Tp, \_Tp, \_Abi>, typename \_R = \_Math\_return\_type\_t< decltype(std::fmod(declval<double>(), \_Arg2::declval())), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::fmod** (const simd< \_Tp, \_Abi > &\_\_x, const typename \_Arg2::type &\_\_y)
- constexpr float **std::fmod** (float \_\_x, float \_\_y)
- constexpr long double **std::fmod** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, int >::\_\_type **std::fpclassify** (\_Tp \_\_x)
- constexpr int **std::fpclassify** (double \_\_x)
- constexpr int **std::fpclassify** (float \_\_x)
- constexpr int **std::fpclassify** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::frexp** (\_Tp \_\_x, int \*\_\_exp)

- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>`  
`enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > std::frexp (const simd< _Tp, _Abi > &__x,`  
`_Samesize< int, simd< _Tp, _Abi > > *__exp)`
- `float std::frexp (float __x, int *__exp)`
- `long double std::frexp (long double __x, int *__exp)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type std::hypot (_Tp __x, _Up __y)`
- `template<typename _Tp, typename _Abi>`  
`_GLIBCXX_SIMD_INTRINSIC simd< _Tp, _Abi > std::hypot (const simd< _Tp, _Abi > &__x, const simd<`  
`_Tp, _Abi > &__y)`
- `constexpr float std::hypot (float __x, float __y)`
- `constexpr long double std::hypot (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type std::ilogb (_Tp __x)`
- `constexpr int std::ilogb (float __x)`
- `constexpr int std::ilogb (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::ilogb(declval<double>())), _Tp,`  
`_Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::ilogb (simd< _Tp,`  
`_Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type std::isfinite (_Tp)`
- `constexpr bool std::isfinite (double __x)`
- `constexpr bool std::isfinite (float __x)`
- `constexpr bool std::isfinite (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if< (__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value),`  
`bool >::__type std::isgreater (_Tp __x, _Up __y)`
- `constexpr bool std::isgreater (double __x, double __y)`
- `constexpr bool std::isgreater (float __x, float __y)`
- `constexpr bool std::isgreater (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if< (__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value),`  
`bool >::__type std::isgreaterequal (_Tp __x, _Up __y)`
- `constexpr bool std::isgreaterequal (double __x, double __y)`
- `constexpr bool std::isgreaterequal (float __x, float __y)`
- `constexpr bool std::isgreaterequal (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type std::isinf (_Tp)`
- `constexpr bool std::isinf (double __x)`
- `constexpr bool std::isinf (float __x)`
- `constexpr bool std::isinf (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if< (__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value),`  
`bool >::__type std::isless (_Tp __x, _Up __y)`
- `constexpr bool std::isless (double __x, double __y)`
- `constexpr bool std::isless (float __x, float __y)`
- `constexpr bool std::isless (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if< (__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value),`  
`bool >::__type std::islessequal (_Tp __x, _Up __y)`
- `constexpr bool std::islessequal (double __x, double __y)`

- constexpr bool **std::islessequal** (float \_\_x, float \_\_y)
- constexpr bool **std::islessequal** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<(\_is\_arithmetic< \_Tp >::\_\_value && \_is\_arithmetic< \_Up >::\_\_value),  
bool >::\_\_type **std::islessgreater** (\_Tp \_\_x, \_Up \_\_y)
- constexpr bool **std::islessgreater** (double \_\_x, double \_\_y)
- constexpr bool **std::islessgreater** (float \_\_x, float \_\_y)
- constexpr bool **std::islessgreater** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_is\_integer< \_Tp >::\_\_value, bool >::\_\_type **std::isnan** (\_Tp)
- constexpr bool **std::isnan** (double \_\_x)
- constexpr bool **std::isnan** (float \_\_x)
- constexpr bool **std::isnan** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_is\_integer< \_Tp >::\_\_value, bool >::\_\_type **std::isnormal** (\_Tp \_\_x)
- constexpr bool **std::isnormal** (double \_\_x)
- constexpr bool **std::isnormal** (float \_\_x)
- constexpr bool **std::isnormal** (long double \_\_x)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<(\_is\_arithmetic< \_Tp >::\_\_value && \_is\_arithmetic< \_Up >::\_\_value),  
bool >::\_\_type **std::isunordered** (\_Tp \_\_x, \_Up \_\_y)
- constexpr bool **std::isunordered** (double \_\_x, double \_\_y)
- constexpr bool **std::isunordered** (float \_\_x, float \_\_y)
- constexpr bool **std::isunordered** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::ldexp** (\_Tp \_\_x, int  
\_\_exp)
- template<typename \_Tp, typename \_Abi, typename..., typename \_Arg2 = \_Extra\_argument\_type<int, \_Tp, \_Abi>, typename \_R = \_Math\_return\_type\_t< decltype(std::ldexp(declval<double>()), \_Arg2::declval()), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE **enable\_if\_t**< is\_floating\_point\_v< \_Tp >, \_R > **std::ldexp** (const simd<  
\_Tp, \_Abi > &\_\_x, const typename \_Arg2::type &\_\_y)
- constexpr float **std::ldexp** (float \_\_x, int \_\_exp)
- constexpr long double **std::ldexp** (long double \_\_x, int \_\_exp)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::lgamma** (\_Tp \_\_x)
- constexpr float **std::lgamma** (float \_\_x)
- constexpr long double **std::lgamma** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::lgamma(declval<double>())),  
\_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE **enable\_if\_t**< is\_floating\_point\_v< \_Tp >, \_R > **std::lgamma** (simd< \_Tp,  
\_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_is\_integer< \_Tp >::\_\_value, longlong >::\_\_type **std::llrint** (\_Tp \_\_x)
- constexpr long long **std::llrint** (float \_\_x)
- constexpr long long **std::llrint** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::llrint(declval<double>())), \_Tp,  
\_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE **enable\_if\_t**< is\_floating\_point\_v< \_Tp >, \_R > **std::llrint** (simd< \_Tp, <  
\_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_is\_integer< \_Tp >::\_\_value, longlong >::\_\_type **std::llround** (\_Tp \_\_x)
- constexpr long long **std::llround** (float \_\_x)

- constexpr long long **std::llround** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::llround(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::llround** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log** (\_Tp \_\_x)
- constexpr float **std::log** (float \_\_x)
- constexpr long double **std::log** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::log(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::log** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log10** (\_Tp \_\_x)
- constexpr float **std::log10** (float \_\_x)
- constexpr long double **std::log10** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::log10(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::log10** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log1p** (\_Tp \_\_x)
- constexpr float **std::log1p** (float \_\_x)
- constexpr long double **std::log1p** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::log1p(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::log1p** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log2** (\_Tp \_\_x)
- constexpr float **std::log2** (float \_\_x)
- constexpr long double **std::log2** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::log2(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::log2** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::logb** (\_Tp \_\_x)
- template<typename \_Tp, typename \_Abi, typename = \_\_detail::\_\_odr\_helper>  
[enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >::value, simd< \_Tp, \_Abi > > **std::logb** (const simd< \_Tp, \_Abi > &\_\_x)
- constexpr float **std::logb** (float \_\_x)
- constexpr long double **std::logb** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, long >::\_\_type **std::lrint** (\_Tp \_\_x)
- constexpr long **std::lrint** (float \_\_x)
- constexpr long **std::lrint** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename..., typename \_R = \_Math\_return\_type\_t< decltype(std::lrint(declval<double>()))>, \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::lrint** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, long >::\_\_type **std::lround** (\_Tp \_\_x)

- constexpr long **std::lround** (float \_\_x)
- constexpr long **std::lround** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename... , typename \_R = \_Math\_return\_type\_t< decltype(std::lround(declval<double>())),  $\leftrightarrow$  \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::lround** (simd< \_Tp, \_Abi > \_\_x)
- template<typename \_Tp, typename \_Abi, typename = \_\_detail::\_\_odr\_helper>  
[enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, simd< \_Tp, \_Abi > > **std::modf** (const simd< \_Tp, \_Abi > &\_\_x, simd< \_Tp, \_Abi > \*\_\_iptr)
- float **std::modf** (float \_\_x, float \*\_\_iptr)
- long double **std::modf** (long double \_\_x, long double \*\_\_iptr)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::nearbyint** (\_Tp \_\_x)
- constexpr float **std::nearbyint** (float \_\_x)
- constexpr long double **std::nearbyint** (long double \_\_x)
- template<typename \_Tp, typename \_Abi, typename... , typename \_R = \_Math\_return\_type\_t< decltype(std::nearbyint(declval<double>())),  $\leftrightarrow$  \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::nearbyint** (simd<  $\leftrightarrow$  \_Tp, \_Abi > \_\_x)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::nextafter** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp, typename \_Abi, typename... , typename \_Arg2 = \_Extra\_argument\_type< \_Tp, \_Tp, \_Abi>, typename \_R =  $\leftrightarrow$  \_Math\_return\_type\_t< decltype(std::nextafter(declval<double>(), \_Arg2::declval())), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::nextafter** (const simd< \_Tp, \_Abi > &\_\_x, const typename \_Arg2::type &\_\_y)
- constexpr float **std::nextafter** (float \_\_x, float \_\_y)
- constexpr long double **std::nextafter** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::nexttoward** (\_Tp  $\leftrightarrow$  \_\_x, long double \_\_y)
- constexpr float **std::nexttoward** (float \_\_x, long double \_\_y)
- constexpr long double **std::nexttoward** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::pow** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp, typename \_Abi, typename... , typename \_Arg2 = \_Extra\_argument\_type< \_Tp, \_Tp, \_Abi>, typename \_R =  $\leftrightarrow$  \_Math\_return\_type\_t< decltype(std::pow(declval<double>(), \_Arg2::declval())), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::pow** (const simd< \_Tp, \_Abi > &\_\_x, const typename \_Arg2::type &\_\_y)
- constexpr float **std::pow** (float \_\_x, float \_\_y)
- constexpr long double **std::pow** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::remainder** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp, typename \_Abi, typename... , typename \_Arg2 = \_Extra\_argument\_type< \_Tp, \_Tp, \_Abi>, typename \_R =  $\leftrightarrow$  \_Math\_return\_type\_t< decltype(std::remainder(declval<double>(), \_Arg2::declval())), \_Tp, \_Abi>>  
\_GLIBCXX\_SIMD\_ALWAYS\_INLINE [enable\\_if\\_t](#)< is\_floating\_point\_v< \_Tp >, \_R > **std::remainder** (const simd< \_Tp, \_Abi > &\_\_x, const typename \_Arg2::type &\_\_y)
- constexpr float **std::remainder** (float \_\_x, float \_\_y)
- constexpr long double **std::remainder** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::remquo** (\_Tp \_\_x, \_Up \_\_y, int \*\_\_pquo)

- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _Arg3 = _Extra_argument_type<int*, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::remquo(declval<double>()), _Arg2::declval(), _Arg3::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v<_Tp>, _R> std::remquo (const simd<_Tp, _Abi> &__x, const typename _Arg2::type &__y, const typename _Arg3::type &__z)`
- `float std::remquo (float __x, float __y, int *__pquo)`
- `long double std::remquo (long double __x, long double __y, int *__pquo)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer<_Tp>::__value, double>::__type std::rint (_Tp __x)`
- `constexpr float std::rint (float __x)`
- `constexpr long double std::rint (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::rint(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v<_Tp>, _R> std::rint (simd<_Tp, _Abi> &__x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer<_Tp>::__value, double>::__type std::round (_Tp __x)`
- `constexpr float std::round (float __x)`
- `constexpr long double std::round (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::round(declval<double>())), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v<_Tp>, _R> std::round (simd<_Tp, _Abi> &__x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer<_Tp>::__value, double>::__type std::scalbn (_Tp __x, long __ex)`
- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<long, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::scalbn(declval<double>()), _Arg2::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v<_Tp>, _R> std::scalbn (const simd<_Tp, _Abi> &__x, const typename _Arg2::type &__y)`
- `constexpr float std::scalbn (float __x, long __ex)`
- `constexpr long double std::scalbn (long double __x, long __ex)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer<_Tp>::__value, double>::__type std::scalbn (_Tp __x, int __ex)`
- `template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<int, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::scalbn(declval<double>()), _Arg2::declval()), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v<_Tp>, _R> std::scalbn (const simd<_Tp, _Abi> &__x, const typename _Arg2::type &__y)`
- `constexpr float std::scalbn (float __x, int __ex)`
- `constexpr long double std::scalbn (long double __x, int __ex)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer<_Tp>::__value, bool>::__type std::signbit (_Tp __x)`
- `constexpr bool std::signbit (double __x)`
- `constexpr bool std::signbit (float __x)`
- `constexpr bool std::signbit (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer<_Tp>::__value, double>::__type std::sin (_Tp __x)`
- `template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>`  
`enable_if_t< is_floating_point_v<_Tp>, simd<_Tp, _Abi>> std::sin (const simd<_Tp, _Abi> &__x)`
- `constexpr float std::sin (float __x)`
- `constexpr long double std::sin (long double __x)`



- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::sinh (_Tp __x)`
- `constexpr float std::sinh (float __x)`
- `constexpr long double std::sinh (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::sinh(declval<double>()))), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::sinh (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::sqrt (_Tp __x)`
- `constexpr float std::sqrt (float __x)`
- `constexpr long double std::sqrt (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::sqrt(declval<double>()))), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::sqrt (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::tan (_Tp __x)`
- `constexpr float std::tan (float __x)`
- `constexpr long double std::tan (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tan(declval<double>()))), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::tan (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::tanh (_Tp __x)`
- `constexpr float std::tanh (float __x)`
- `constexpr long double std::tanh (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tanh(declval<double>()))), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::tanh (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::tgamma (_Tp __x)`
- `constexpr float std::tgamma (float __x)`
- `constexpr long double std::tgamma (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tgamma(declval<double>()))), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::tgamma (simd< _Tp, _Abi > __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::trunc (_Tp __x)`
- `constexpr float std::trunc (float __x)`
- `constexpr long double std::trunc (long double __x)`
- `template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::trunc(declval<double>()))), _Tp, _Abi>>`  
`_GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > std::trunc (simd< _Tp, _Abi > __x)`



### 6.218.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.219 `ext/cmath` File Reference

### Namespaces

- namespace `__gnu_cxx`

### Macros

- `#define __EXT_CMATH`

### Variables

- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__gamma_e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__ln_10`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__ln_2`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__ln_3`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__log10_e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__log2_e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__one_div_e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__one_div_pi`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__one_div_root_2`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__phi`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__pi`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__pi_half`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__pi_quarter`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__pi_third`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_2`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_3`

- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_5`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_7`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_pi_div_2`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__two_div_pi`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__two_div_root_pi`

### 6.219.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.220 tr1/cmath File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CMATH`

### Functions

- `template<typename _Tp>`  
`__gnu_cxx::__promote<_Tp>::__type std::tr1::assoc\_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::tr1::assoc\_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::tr1::assoc\_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote<_Tp>::__type std::tr1::assoc\_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float std::tr1::assoc\_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::tr1::assoc\_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tpx, typename _Tpy>`  
`__gnu_cxx::__promote_2<_Tpx, _Tpy>::__type std::tr1::beta (_Tpx __x, _Tpy __y)`
- `float std::tr1::betaf (float __x, float __y)`
- `long double std::tr1::betal (long double __x, long double __y)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote<_Tp>::__type std::tr1::comp\_ellint\_1 (_Tp __k)`
- `float std::tr1::comp\_ellint\_1f (float __k)`
- `long double std::tr1::comp\_ellint\_1l (long double __k)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote<_Tp>::__type std::tr1::comp\_ellint\_2 (_Tp __k)`
- `float std::tr1::comp\_ellint\_2f (float __k)`
- `long double std::tr1::comp\_ellint\_2l (long double __k)`
- `template<typename _Tp, typename _Tpn>`  
`__gnu_cxx::__promote_2<_Tp, _Tpn>::__type std::tr1::comp\_ellint\_3 (_Tp __k, _Tpn __nu)`
- `float std::tr1::comp\_ellint\_3f (float __k, float __nu)`
- `long double std::tr1::comp\_ellint\_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type std::tr1::conf\_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`

- float [std::tr1::conf\\_hypergf](#) (float \_\_a, float \_\_c, float \_\_x)
- long double [std::tr1::conf\\_hypergl](#) (long double \_\_a, long double \_\_c, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type [std::tr1::cyl\\_bessel\\_i](#) (\_Tpnu \_\_nu, \_Tp \_\_x)
- float [std::tr1::cyl\\_bessel\\_if](#) (float \_\_nu, float \_\_x)
- long double [std::tr1::cyl\\_bessel\\_il](#) (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type [std::tr1::cyl\\_bessel\\_j](#) (\_Tpnu \_\_nu, \_Tp \_\_x)
- float [std::tr1::cyl\\_bessel\\_jf](#) (float \_\_nu, float \_\_x)
- long double [std::tr1::cyl\\_bessel\\_jl](#) (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type [std::tr1::cyl\\_bessel\\_k](#) (\_Tpnu \_\_nu, \_Tp \_\_x)
- float [std::tr1::cyl\\_bessel\\_kf](#) (float \_\_nu, float \_\_x)
- long double [std::tr1::cyl\\_bessel\\_kl](#) (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type [std::tr1::cyl\\_neumann](#) (\_Tpnu \_\_nu, \_Tp \_\_x)
- float [std::tr1::cyl\\_neumannf](#) (float \_\_nu, float \_\_x)
- long double [std::tr1::cyl\\_neumannl](#) (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type [std::tr1::ellint\\_1](#) (\_Tp \_\_k, \_Tpp \_\_phi)
- float [std::tr1::ellint\\_1f](#) (float \_\_k, float \_\_phi)
- long double [std::tr1::ellint\\_1l](#) (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type [std::tr1::ellint\\_2](#) (\_Tp \_\_k, \_Tpp \_\_phi)
- float [std::tr1::ellint\\_2f](#) (float \_\_k, float \_\_phi)
- long double [std::tr1::ellint\\_2l](#) (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type [std::tr1::ellint\\_3](#) (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float [std::tr1::ellint\\_3f](#) (float \_\_k, float \_\_nu, float \_\_phi)
- long double [std::tr1::ellint\\_3l](#) (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::tr1::expint](#) (\_Tp \_\_x)
- float [std::tr1::expintf](#) (float \_\_x)
- long double [std::tr1::expintl](#) (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::tr1::fabs](#) (\_Tp \_\_x)
- float [std::tr1::fabs](#) (float \_\_x)
- long double [std::tr1::fabs](#) (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::tr1::hermite](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::tr1::hermitef](#) (unsigned int \_\_n, float \_\_x)
- long double [std::tr1::hermitel](#) (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_4< \_Tpa, \_Tpb, \_Tpc, \_Tp >::\_\_type [std::tr1::hyperg](#) (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float [std::tr1::hypergf](#) (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double [std::tr1::hypergl](#) (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::tr1::laguerre](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::tr1::laguerref](#) (unsigned int \_\_n, float \_\_x)
- long double [std::tr1::laguerrel](#) (unsigned int \_\_n, long double \_\_x)

- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::legendre` (unsigned int \_\_n, \_Tp \_\_x)
- float `std::tr1::legendref` (unsigned int \_\_n, float \_\_x)
- long double `std::tr1::legendrel` (unsigned int \_\_n, long double \_\_x)
- `template<typename _Tp, typename _Up>`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type std::tr1::pow` (\_Tp \_\_x, \_Up \_\_y)
- float `std::tr1::pow` (float \_\_x, float \_\_y)
- long double `std::tr1::pow` (long double \_\_x, long double \_\_y)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::riemann_zeta` (\_Tp \_\_x)
- float `std::tr1::riemann_zetaf` (float \_\_x)
- long double `std::tr1::riemann_zetal` (long double \_\_x)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel` (unsigned int \_\_n, \_Tp \_\_x)
- float `std::tr1::sph_besself` (unsigned int \_\_n, float \_\_x)
- long double `std::tr1::sph_bessell` (unsigned int \_\_n, long double \_\_x)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float `std::tr1::sph_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double `std::tr1::sph_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann` (unsigned int \_\_n, \_Tp \_\_x)
- float `std::tr1::sph_neumannf` (unsigned int \_\_n, float \_\_x)
- long double `std::tr1::sph_neumannl` (unsigned int \_\_n, long double \_\_x)

### 6.220.1 Detailed Description

This is a TR1 C++ Library header.

## 6.221 codecvt File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_CODECVT`
- `#define _GLIBCXX_CODECVT_SPECIALIZATION(_NAME, _ELEM)`
- `#define _GLIBCXX_CODECVT_SPECIALIZATION2(_NAME, _ELEM)`

### Enumerations

- enum `codecvt_mode` { `consume_header` , `generate_header` , `little_endian` }

### 6.221.1 Detailed Description

This is a Standard C++ Library header.

## 6.222 complex File Reference

### Classes

- class `std::complex< _Tp >`
- class `std::complex< double >`
- class `std::complex< float >`
- class `std::complex< long double >`

### Namespaces

- namespace `std`

### Macros

- `#define __glibcxx_want_complex_udls`
- `#define __glibcxx_want_constexpr_complex`
- `#define __glibcxx_want_tuple_like`
- `#define _GLIBCXX26_DECLARE_COMPLEX_TUPLE_HELPER_ACCESSOR`
- `#define _GLIBCXX_COMPLEX`

### Functions

- `template<typename _Tp>`  
`_Tp std::__complex_abs (const complex< _Tp > &__z)`
- `__complex__ double std::__complex_acos (__complex__ double __z)`
- `__complex__ float std::__complex_acos (__complex__ float __z)`
- `__complex__ long double std::__complex_acos (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_acos (const std::complex< _Tp > &__z)`
- `__complex__ double std::__complex_acosh (__complex__ double __z)`
- `__complex__ float std::__complex_acosh (__complex__ float __z)`
- `__complex__ long double std::__complex_acosh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`_Tp std::__complex_arg (const complex< _Tp > &__z)`
- `__complex__ double std::__complex_asin (__complex__ double __z)`
- `__complex__ float std::__complex_asin (__complex__ float __z)`
- `__complex__ long double std::__complex_asin (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_asin (const std::complex< _Tp > &__z)`
- `__complex__ double std::__complex_asinh (__complex__ double __z)`
- `__complex__ float std::__complex_asinh (__complex__ float __z)`
- `__complex__ long double std::__complex_asinh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_asinh (const std::complex< _Tp > &__z)`
- `__complex__ double std::__complex_atan (__complex__ double __z)`
- `__complex__ float std::__complex_atan (__complex__ float __z)`
- `__complex__ long double std::__complex_atan (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_atan (const std::complex< _Tp > &__z)`
- `__complex__ double std::__complex_atanh (__complex__ double __z)`

- `__complex__ float std::__complex_atanh (__complex__ float __z)`
- `__complex__ long double std::__complex_atanh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `complex< double > std::__complex_proj (const complex< double > &__z)`
- `complex< float > std::__complex_proj (const complex< float > &__z)`
- `complex< long double > std::__complex_proj (const complex< long double > &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::acos (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::acosh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::arg (_Tp __x)`
- `template<typename _Tp>`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::asin (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::asinh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::atan (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::atanh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::conj (_Tp __x)`

- `template<typename _Tp>`  
`constexpr complex<_Tp> std::conj (const complex<_Tp> &)`
- `template<typename _Tp>`  
`complex<_Tp> std::cos (const complex<_Tp> &)`
- `template<typename _Tp>`  
`complex<_Tp> std::cosh (const complex<_Tp> &)`
- `template<typename _Tp>`  
`complex<_Tp> std::exp (const complex<_Tp> &)`
- `template<typename _Tp>`  
`_Tp std::fabs (const std::complex<_Tp> &__z)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote<_Tp>::__type std::imag (_Tp)`
- `template<typename _Tp>`  
`constexpr _Tp std::imag (const complex<_Tp> &__z)`
- `template<typename _Tp>`  
`complex<_Tp> std::log (const complex<_Tp> &)`
- `template<typename _Tp>`  
`complex<_Tp> std::log10 (const complex<_Tp> &)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote<_Tp>::__type std::norm (_Tp __x)`
- `template<typename _Tp>`  
`_Tp constexpr std::norm (const complex<_Tp> &)`
- `template<typename _Tp>`  
`constexpr _Tp std::norm (const complex<_Tp> &__z)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator+ (const complex<_Tp> &__x)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator- (const complex<_Tp> &__x)`
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &__os, const complex<_Tp> &__x)`
- `template ostream & std::operator<< (ostream &, const complex<double> &)`
- `template ostream & std::operator<< (ostream &, const complex<float> &)`
- `template ostream & std::operator<< (ostream &, const complex<long double> &)`
- `template wostream & std::operator<< (wostream &, const complex<double> &)`
- `template wostream & std::operator<< (wostream &, const complex<float> &)`
- `template wostream & std::operator<< (wostream &, const complex<long double> &)`
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic_istream<_CharT, _Traits> & std::operator>> (basic_istream<_CharT, _Traits> &__is, complex<_Tp> &__x)`
- `template istream & std::operator>> (istream &, complex<double> &)`
- `template istream & std::operator>> (istream &, complex<float> &)`
- `template istream & std::operator>> (istream &, complex<long double> &)`
- `template wistream & std::operator>> (wistream &, complex<double> &)`
- `template wistream & std::operator>> (wistream &, complex<float> &)`
- `template wistream & std::operator>> (wistream &, complex<long double> &)`
- `template<typename _Tp>`  
`complex<_Tp> std::polar (const _Tp &, const _Tp &=__Tp(0))`
- `template<typename _Tp>`  
`complex<_Tp> std::pow (const _Tp &, const complex<_Tp> &)`
- `template<typename _Tp, typename _Up>`  
`std::complex<typename __gnu_cxx::__promote_2<_Tp, _Up>::__type> std::pow (const _Tp &__x, const std::complex<_Up> &__y)`

- `template<typename _Tp>`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp>`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::pow (const complex< _Tp > &, int)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::proj (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::proj (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote< _Tp >::__type std::real (_Tp __x)`
- `template<typename _Tp>`  
`constexpr _Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
  
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`



- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator\* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`

### 6.222.1 Detailed Description

This is a Standard C++ Library header.

## 6.223 `tr1/complex` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define \_GLIBCXX\_TR1\_COMPLEX`

### Functions

- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::acos (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`_Tp std::tr1::arg (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::tr1::conj (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`

- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`constexpr _Tp std::tr1::imag (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`_Tp constexpr std::tr1::norm (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::tr1::polar (const _Tp &, const _Tp &=_Tp(0))`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::polar (const _Tp &__rho,`  
`const _Up &__theta)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const _Tp &__x,`  
`const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex<`  
`_Tp > &__x, const _Up &__y)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex<`  
`_Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`constexpr _Tp std::tr1::real (const complex< _Tp > &__z)`

### 6.223.1 Detailed Description

This is a TR1 C++ Library header.

## 6.224 complex.h File Reference

### 6.224.1 Detailed Description

This is a Standard C++ Library header.

## 6.225 concepts File Reference

### Macros

- `#define __glibcxx_want_concepts`
- `#define _GLIBCXX_CONCEPTS`

### 6.225.1 Detailed Description

This is a Standard C++ Library header.

## 6.226 condition\_variable File Reference

### Classes

- class `std::condition_variable`
- class `std::condition_variable_any`

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CONDITION_VARIABLE`

### Enumerations

- enum class [std::cv\\_status](#) { `no_timeout` , `timeout` }

### Functions

- void `std::notify_all_at_thread_exit` ([condition\\_variable](#) &, [unique\\_lock](#)< [mutex](#) >)

#### 6.226.1 Detailed Description

This is a Standard C++ Library header.

## 6.227 csetjmp File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSETJMP`
- `#define setjmp(env)`

#### 6.227.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.228 csignal File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSIGNAL`

#### 6.228.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.229 `cstdalign` File Reference

### Macros

- `#define _GLIBCXX_CSTDALIGN`

### 6.229.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

## 6.230 `cstdarg` File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_CSTDARG`
- `#define va_end(ap)`

### 6.230.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.231 `tr1/cstdarg` File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDARG`

### 6.231.1 Detailed Description

This is a TR1 C++ Library header.

## 6.232 `cstdbool` File Reference

### Macros

- `#define _GLIBCXX_CSTDBOOL`

### 6.232.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

## 6.233 tr1/cstdbool File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDBOOL`

### 6.233.1 Detailed Description

This is a TR1 C++ Library header.

## 6.234 cstdint File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_byte`
- `#define _GLIBCXX_CSTDDEF`

### 6.234.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdint.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.235 cstdint File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSTDINT`

### 6.235.1 Detailed Description

This is a Standard C++ Library header.

## 6.236 tr1/cstdint File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CSTDINT`

### 6.236.1 Detailed Description

This is a TR1 C++ Library header.

## 6.237 `cstdio` File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_CSTDIO`

#### 6.237.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.238 `tr1/cstdio` File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDIO`

#### 6.238.1 Detailed Description

This is a TR1 C++ Library header.

## 6.239 `cstdlib` File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_CSTDLIB`
- `#define _GLIBCXX_INCLUDE_NEXT_C_HEADERS`

### Functions

- `ldiv_t std::div (long __i, long __j) noexcept`

#### 6.239.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.240 `tr1/cstdlib` File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDLIB`

### 6.240.1 Detailed Description

This is a TR1 C++ Library header.

## 6.241 cstring File Reference

### Namespaces

- namespace `std`

### Macros

- `#define __glibcxx_want_freestanding_cstring`
- `#define _GLIBCXX_CSTRING`

### Functions

- `void * std::memchr (void *__s, int __c, size_t __n)`
- `char * std::strchr (char *__s, int __n)`
- `char * std::strpbrk (char *__s1, const char *__s2)`
- `char * std::strrchr (char *__s, int __n)`
- `char * std::strstr (char *__s1, const char *__s2)`

### 6.241.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.242 ctgmath File Reference

### Macros

- `#define _GLIBCXX_CTGMATH`

### 6.242.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

## 6.243 tr1/ctgmath File Reference

### Macros

- `#define _GLIBCXX_TR1_CTGMATH`

### 6.243.1 Detailed Description

This is a TR1 C++ Library header.

## 6.244 ctime File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CTIME`

#### 6.244.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.245 tr1/ctime File Reference

### Macros

- `#define _GLIBCXX_TR1_CTIME`

#### 6.245.1 Detailed Description

This is a TR1 C++ Library header.

## 6.246 cuchar File Reference

### Macros

- `#define _GLIBCXX_CUCHAR`

#### 6.246.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `uchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.247 wchar File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CWCHAR`

### Functions

- `wchar_t * std::wcschr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcpbrk (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wcsrchr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcsstr (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wmemchr (wchar_t *__p, wchar_t __c, size_t __n)`



### 6.247.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.248 tr1/cwchar File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CWCHAR`

### 6.248.1 Detailed Description

This is a TR1 C++ Library header.

## 6.249 cwctype File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CWCTYPE`

### 6.249.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.250 tr1/cwctype File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CWCTYPE`

### 6.250.1 Detailed Description

This is a TR1 C++ Library header.

## 6.251 assertions.h File Reference

### Macros

- `#define __glibcxx_requires_non_empty_range(_First, _Last)`
- `#define __glibcxx_requires_nonempty()`
- `#define __glibcxx_requires_subscript(_N)`
- `#define _GLIBCXX_DEBUG_ASSERT(_Condition)`
- `#define _GLIBCXX_DEBUG_ONLY(_Statement)`
- `#define _GLIBCXX_DEBUG_PEDASSERT(_Condition)`

#### 6.251.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.252 debug.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Macros

- `#define __glibcxx_requires_can_decrement_range(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_can_increment(_First, _Size)`
- `#define __glibcxx_requires_can_increment_range(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_cond(_Cond, _Msg)`
- `#define __glibcxx_requires_heap(_First, _Last)`
- `#define __glibcxx_requires_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_irreflexive(_First, _Last)`
- `#define __glibcxx_requires_irreflexive2(_First, _Last)`
- `#define __glibcxx_requires_irreflexive_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_irreflexive_pred2(_First, _Last, _Pred)`
- `#define __glibcxx_requires_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_sorted(_First, _Last)`
- `#define __glibcxx_requires_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_requires_string(_String)`
- `#define __glibcxx_requires_string_len(_String, _Len)`
- `#define __glibcxx_requires_valid_range(_First, _Last)`

#### 6.252.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.253 formatter.h File Reference

### Classes

- class [\\_\\_gnu\\_debug::type\\_info](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)

### Macros

- `#define _GLIBCXX_TYPEID(_Type)`

### Enumerations

- enum `_Debug_msg_id` {  
    `__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,  
    `__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,  
    `__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,  
    `__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`, `__msg_bad_bitset_flip`,  
    `__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,  
    `__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,  
    `__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,  
    `__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,  
    `__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`, `__msg_distance_bad`,  
    `__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,  
    `__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`, `__msg_erase_after_bad`,  
    `__msg_valid_range2`, `__msg_local_iter_compare_bad`, `__msg_non_empty_range`, `__msg_self_move`↵  
    `__assign`,  
    `__msg_bucket_index_oob`, `__msg_valid_load_factor`, `__msg_equal_allocs`, `__msg_insert_range`↵  
    `from_self`,  
    `__msg_irreflexive_ordering` }

### Functions

- `template<typename _Iterator>`  
    `constexpr bool __gnu_debug::__check_singular (_Iterator const &)`

#### 6.253.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.254 functions.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

- `template<typename _ForwardIterator, typename _Tp>`  
    `constexpr bool __gnu_debug::__check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last,`  
    `const _Tp &__value)`

- `template<typename _ForwardIterator, typename _Tp, typename _Pred>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_partitioned\_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted (const _InputIterator &__first, const _InputIterator &__last, \_Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_aux (_ForwardIterator __first, _ForwardIterator __last, \_Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _ForwardIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _InputIterator &, const _InputIterator &, \_Predicate, std::input\_iterator\_tag)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, \_Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &, const _InputIterator &, \_Predicate, std::\_\_false\_type)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &, const _InputIterator &, std::\_\_false\_type)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &__first, const _InputIterator &__last, \_Predicate __pred, std::\_\_true\_type)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &__first, const _InputIterator &__last, std::\_\_true\_type)`
- `template<typename _InputIterator>`  
`\_InputIterator \_\_gnu\_debug::\_\_check\_valid\_range (const \_InputIterator &__first, const \_InputIterator &__last, const char *__file, unsigned int __line, const char *__function)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool \_\_gnu\_debug::\_\_foreign\_iterator (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &__it, \_InputIterator __other, \_InputIterator __other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Integral>`  
`bool \_\_gnu\_debug::\_\_foreign\_iterator\_aux (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, \_Integral, \_Integral, std::\_\_true\_type)`

- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __gnu_debug::__foreign_iterator_aux (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`_InputIterator __other, _InputIterator __other_end, std::__false_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, typename`  
`_OtherCategory>`  
`bool __gnu_debug::__foreign_iterator_aux2 (const __Safe_iterator< _Iterator, _Sequence, _Category > &, const`  
`__Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &, const __Safe_iterator< _OtherIterator,`  
`_OtherSequence, _OtherCategory > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __gnu_debug::__foreign_iterator_aux2 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`const _InputIterator &__other, const _InputIterator &__other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>`  
`bool __gnu_debug::__foreign_iterator_aux2 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`const __Safe_iterator< _OtherIterator, _Sequence, _Category > &__other, const __Safe_iterator< _OtherIterator,`  
`_Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __gnu_debug::__foreign_iterator_aux3 (const __Safe_iterator< _Iterator, _Sequence, _Category > &,`  
`const _InputIterator &, const _InputIterator &, std::__false_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __gnu_debug::__foreign_iterator_aux3 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`const _InputIterator &__other, const _InputIterator &__other_end, std::__true_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool __gnu_debug::__foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence, _Category > &,...)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool __gnu_debug::__foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`const typename _Sequence::value_type *__other)`
- `template<typename _Iterator>`  
`constexpr bool __gnu_debug::__is_irreflexive (_Iterator __it)`
- `template<typename _Iterator, typename _Pred>`  
`constexpr bool __gnu_debug::__is_irreflexive_pred (_Iterator __it, _Pred __pred)`

### 6.254.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.255 helper\_functions.h File Reference

### Namespaces

- namespace `__gnu_debug`

### Enumerations

- enum `__gnu_debug::Distance_precision` {  
`__dp_none` , `__dp_equality` , `__dp_sign` , `__dp_sign_max_size` ,  
`__dp_exact` }

### Functions

- `template<typename _Iterator>`  
`constexpr _Iterator __gnu_debug::__base (_Iterator __it)`
- `template<typename _InputIterator, typename _Size>`  
`constexpr bool __gnu_debug::__can_advance (_InputIterator, _Size)`

- `template<typename _InputIterator, typename _Diff>`  
`constexpr bool \_\_gnu\_debug::\_\_can\_advance (_InputIterator, const std::pair< _Diff, \_Distance\_precision > &, int)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>`  
`bool \_\_gnu\_debug::\_\_can\_advance (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, _Size)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>`  
`bool \_\_gnu\_debug::\_\_can\_advance (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, const std::pair< _Diff, \_Distance\_precision > &, int)`
- `template<typename _Iterator>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_singular (_Iterator const &)`
- `template<typename _Tp>`  
`constexpr bool \_\_gnu\_debug::\_\_check\_singular (_Tp *const &__ptr)`
- `bool \_\_gnu\_debug::\_\_check\_singular\_aux (const class \_Safe\_iterator\_base *)`
- `bool \_\_gnu\_debug::\_\_check\_singular\_aux (const void *)`
- `template<typename _Iterator>`  
`constexpr \_Distance\_traits< _Iterator >::__type \_\_gnu\_debug::\_\_get\_distance (_Iterator __lhs, _Iterator __rhs)`
- `template<typename _Iterator>`  
`constexpr \_Distance\_traits< _Iterator >::__type \_\_gnu\_debug::\_\_get\_distance (_Iterator __lhs, _Iterator __rhs, std::input\_iterator\_tag)`
- `template<typename _Iterator>`  
`constexpr \_Distance\_traits< _Iterator >::__type \_\_gnu\_debug::\_\_get\_distance (_Iterator __lhs, _Iterator __rhs, std::random\_access\_iterator\_tag)`
- `template<typename _Iterator>`  
`constexpr \_Iterator \_\_gnu\_debug::\_\_unsafe (_Iterator __it)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range (_InputIterator __first, _InputIterator __last, typename \_Distance\_traits< _InputIterator >::__type &__dist)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool \_\_gnu\_debug::\_\_valid\_range (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, const \_Safe\_iterator< _Iterator, _Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool \_\_gnu\_debug::\_\_valid\_range (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, typename \_Distance\_traits< _Iterator >::__type &)`
- `template<typename _Iterator, typename _Sequence>`  
`bool \_\_gnu\_debug::\_\_valid\_range (const \_Safe\_local\_iterator< _Iterator, _Sequence > &, const \_Safe\_local\_iterator< _Iterator, _Sequence > &)`
- `template<typename _Iterator, typename _Sequence>`  
`bool \_\_gnu\_debug::\_\_valid\_range (const \_Safe\_local\_iterator< _Iterator, _Sequence > &, const \_Safe\_local\_iterator< _Iterator, _Sequence > &, typename \_Distance\_traits< _Iterator >::__type &)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_InputIterator __first, _InputIterator __last, std::false\_type)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_InputIterator __first, _InputIterator __last, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_InputIterator __first, _InputIterator __last, typename \_Distance\_traits< _InputIterator >::__type &__dist, std::false\_type)`
- `template<typename _Integral>`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_Integral, _Integral, std::true\_type)`

- `template<typename _Integral>`  
`constexpr bool __gnu_debug::__valid_range_aux ( _Integral, _Integral, typename _Distance_traits< _Integral`  
`>::__type &__dist, std::__true_type)`

### 6.255.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.256 macros.h File Reference

### Macros

- `#define __glibcxx_check_bucket_index(_N)`
- `#define __glibcxx_check_can_decrement_range(_First1, _Last1, _First2)`
- `#define __glibcxx_check_can_increment(_First, _Size)`
- `#define __glibcxx_check_can_increment_dist(_First, _Dist, _Way)`
- `#define __glibcxx_check_can_increment_range(_First1, _Last1, _First2)`
- `#define __glibcxx_check_equal_allocs(_This, _Other)`
- `#define __glibcxx_check_erase(_Position)`
- `#define __glibcxx_check_erase2(_CPosition)`
- `#define __glibcxx_check_erase_after(_Position)`
- `#define __glibcxx_check_erase_range(_First, _Last)`
- `#define __glibcxx_check_erase_range_after(_First, _Last)`
- `#define __glibcxx_check_heap(_First, _Last)`
- `#define __glibcxx_check_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_insert(_Position)`
- `#define __glibcxx_check_insert_after(_Position)`
- `#define __glibcxx_check_insert_range(_Position, _First, _Last, _Dist)`
- `#define __glibcxx_check_insert_range_after(_Position, _First, _Last, _Dist)`
- `#define __glibcxx_check_irreflexive(_First, _Last)`
- `#define __glibcxx_check_irreflexive2(_First, _Last)`
- `#define __glibcxx_check_irreflexive_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_irreflexive_pred2(_First, _Last, _Pred)`
- `#define __glibcxx_check_max_load_factor(_F)`
- `#define __glibcxx_check_non_empty_range(_First, _Last)`
- `#define __glibcxx_check_nonempty()`
- `#define __glibcxx_check_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_sorted(_First, _Last)`
- `#define __glibcxx_check_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_check_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_check_subscript(_N)`
- `#define __glibcxx_check_valid_constructor_range(_First, _Last)`
- `#define __glibcxx_check_valid_range(_First, _Last)`
- `#define __glibcxx_check_valid_range2(_First, _Last, _Dist)`
- `#define __glibcxx_check_valid_range_at(_First, _Last, _File, _Line, _Func)`
- `#define __GLIBCXX_DEBUG_VERIFY(_Cond, _ErrMsg)`
- `#define __GLIBCXX_DEBUG_VERIFY_AT(_Cond, _ErrMsg, _File, _Line)`
- `#define __GLIBCXX_DEBUG_VERIFY_AT_F(_Cond, _ErrMsg, _File, _Line, _Func)`

### 6.256.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 6.256.2 Macro Definition Documentation

#### **`__glibcxx_check_erase`**

```
#define __glibcxx_check_erase(  
    _Position)
```

Verify that we can erase the element referenced by the iterator `_Position`. We can erase the element if the `_Position` iterator is dereferenceable and references this sequence.

#### **`__glibcxx_check_erase_after`**

```
#define __glibcxx_check_erase_after(  
    _Position)
```

Verify that we can erase the element after the iterator `_Position`. We can erase the element if the `_Position` iterator is before a dereferenceable one and references this sequence.

#### **`__glibcxx_check_erase_range`**

```
#define __glibcxx_check_erase_range(  
    _First,  
    _Last)
```

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

#### **`__glibcxx_check_erase_range_after`**

```
#define __glibcxx_check_erase_range_after(  
    _First,  
    _Last)
```

Verify that we can erase the elements in the iterator range `(_First, _Last)`. We can erase the elements if `(_First, _Last)` is a valid iterator range within this sequence.

#### **`__glibcxx_check_heap_pred`**

```
#define __glibcxx_check_heap_pred(  
    _First,  
    _Last,  
    _Pred)
```

Verify that the iterator range `[_First, _Last)` is a heap w.r.t. the predicate `_Pred`.

#### **`__glibcxx_check_insert`**

```
#define __glibcxx_check_insert(  
    _Position)
```

Verify that we can insert into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

#### **`__glibcxx_check_insert_after`**

```
#define __glibcxx_check_insert_after(  
    _Position)
```



Verify that we can insert into *\*this* after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

#### **`__glibcxx_check_insert_range`**

```
#define __glibcxx_check_insert_range(  
    _Position,  
    _First,  
    _Last,  
    _Dist)
```

Verify that we can insert the values in the iterator range `[_First, _Last)` into *\*this* with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

#### **`__glibcxx_check_insert_range_after`**

```
#define __glibcxx_check_insert_range_after(  
    _Position,  
    _First,  
    _Last,  
    _Dist)
```

Verify that we can insert the values in the iterator range `[_First, _Last)` into *\*this* after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

#### **`__glibcxx_check_partitioned_lower`**

```
#define __glibcxx_check_partitioned_lower(  
    _First,  
    _Last,  
    _Value)
```

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value`.

#### **`__glibcxx_check_partitioned_lower_pred`**

```
#define __glibcxx_check_partitioned_lower_pred(  
    _First,  
    _Last,  
    _Value,  
    _Pred)
```

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

#### **`__glibcxx_check_partitioned_upper_pred`**

```
#define __glibcxx_check_partitioned_upper_pred(  
    _First,  
    _Last,  
    _Value,  
    _Pred)
```

Verify that the iterator range [`_First`, `_Last`) is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

### `__glibcxx_check_sorted_pred`

```
#define __glibcxx_check_sorted_pred(
    _First,
    _Last,
    _Pred)
```

Verify that the iterator range [`_First`, `_Last`) is sorted by the predicate `_Pred`.

### `_GLIBCXX_DEBUG_VERIFY_AT_F`

```
#define _GLIBCXX_DEBUG_VERIFY_AT_F(
    _Cond,
    _ErrMsg,
    _File,
    _Line,
    _Func)
```

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

## 6.257 map.h File Reference

### Classes

- class `std::__debug::map<_Key, _Tp, _Compare, _Allocator>`

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Functions

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::map` (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, less<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_val_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::map` (`_InputIterator`, `_InputIterator`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Compare, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::map` (`initializer_list<pair<_Key, _Tp>>`, `_Allocator`) -> `map<_Key, _Tp, less<_Key>, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::map` (`initializer_list<pair<_Key, _Tp>>`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `map<_Key, _Tp, _Compare, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t<pair<const _Key, _Tp>>` `std::__debug::operator<=>` (`const map<_Key, _Tp, _Compare, _Alloc>` & `lhs`, `const map<_Key, _Tp, _Compare, _Alloc>` & `rhs`)

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator==(const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`void std::__debug::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`

### 6.257.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.258 multimap.h File Reference

### Classes

- class `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Functions

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less< __iter_key_t< _InputIterator > >, typename _Allocator = allocator< __iter_key_t< _InputIterator > >, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Compare = less< _Key >, typename _Allocator = allocator< pair< const _Key, _Tp > >, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::multimap (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< pair< const _Key, _Tp > > std::__debug::operator<=> (const multimap< _Key, _Tp, _Compare, _Alloc > &__lhs, const multimap< _Key, _Tp, _Compare, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator==(const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`

### 6.258.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.259 multiset.h File Reference

### Classes

- class [std::\\_\\_debug::multiset<\\_Key, \\_Compare, \\_Allocator>](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::multiset** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `multiset< typename iterator\_traits<_InputIterator>::value_type, less< typename iterator\_traits<_InputIterator>::value_type>, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator\_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator\_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::multiset** (`_InputIterator`, `_InputIterator`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multiset< typename iterator\_traits<_InputIterator>::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::multiset** (`initializer_list<_Key>`, `_Allocator`) -> `multiset<_Key, less<_Key>, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::multiset** (`initializer_list<_Key>`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multiset<_Key, _Compare, _Allocator >`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t<_Key> std::__debug::operator<=> (const multiset<_Key, _Compare, _Alloc> &_lhs, const multiset<_Key, _Compare, _Alloc> &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator== (const multiset<_Key, _Compare, _Allocator> &_lhs, const multiset<_Key, _Compare, _Allocator> &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`void std::__debug::swap (multiset<_Key, _Compare, _Allocator> &_x, multiset<_Key, _Compare, _Allocator> &_y) noexcept(/\*conditional \*/)`

#### 6.259.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.260 safe\_base.h File Reference

### Classes

- class [\\_\\_gnu\\_debug::Safe\\_iterator\\_base](#)
- class [\\_\\_gnu\\_debug::Safe\\_sequence\\_base](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

- bool [\\_\\_gnu\\_debug::\\_\\_check\\_singular\\_aux](#) (const [\\_Safe\\_iterator\\_base](#) \*\_\_x)

### 6.260.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.261 `safe_container.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_container<\\_SafeContainer, \\_Alloc, \\_SafeBase, \\_IsCxx11AllocatorAware >](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### 6.261.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.262 `safe_iterator.h` File Reference

### Classes

- struct [\\_\\_gnu\\_debug::\\_BeforeBeginHelper<\\_Sequence >](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Iterator, \\_Sequence, \\_Category >](#)
- struct [\\_\\_gnu\\_debug::\\_Sequence\\_traits<\\_Sequence >](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Macros

- `#define GLIBCXX20_CONSTEXPR_NON_LITERAL_SCOPE_BEGIN`
- `#define GLIBCXX20_CONSTEXPR_NON_LITERAL_SCOPE_END`
- `#define GLIBCXX_DEBUG_VERIFY_DIST_OPERANDS(_Lhs, _Rhs)`
- `#define GLIBCXX_DEBUG_VERIFY_EQ_OPERANDS(_Lhs, _Rhs)`
- `#define GLIBCXX_DEBUG_VERIFY_OPERANDS(_Lhs, _Rhs, _BadMsgId, _DiffMsgId)`
- `#define GLIBCXX_DEBUG_VERIFY_REL_OPERANDS(_Lhs, _Rhs)`

### Functions

- `template<typename _Iterator, typename _Sequence>`  
`_Iterator __gnu_debug::__base (const \_Safe\_iterator<_Iterator, _Sequence, std::random\_access\_iterator\_tag`  
`> &__it)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>`  
`bool __gnu_debug::__can_advance (const \_Safe\_iterator<_Iterator, _Sequence, _Category > &, _Size)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>`  
`bool __gnu_debug::__can_advance (const \_Safe\_iterator<_Iterator, _Sequence, _Category > &, const`  
`std::pair<_Diff, \_Distance\_precision > &, int)`
- `template<typename _Iterator, typename _Sequence>`  
`_Iterator __gnu_debug::__unsafe (const \_Safe\_iterator<_Iterator, _Sequence > &__it)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool __gnu_debug::__valid_range (const \_Safe\_iterator<_Iterator, _Sequence, _Category > &, const`  
`\_Safe\_iterator<_Iterator, _Sequence, _Category > &)`

- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool \_\_gnu\_debug::\_\_valid\_range (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, const`  
`\_Safe\_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)`

### 6.262.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.263 safe\_iterator.tcc File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)

### Macros

- `#define \_GLIBCXX\_DEBUG\_SAFE\_ITERATOR\_TCC`

### Functions

- `template<bool _IsMove, typename _Il, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > std::copy\_move\_a (_Il, _Il, const`  
`\_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _Ilte, typename _ISeq, typename _ICat, typename _Olte, typename _OSeq, typename _OCat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Olte, _OSeq, _OCat > std::copy\_move\_a (const \_\_gnu\_debug::Safe\_iterator<`  
`_Ilte, _ISeq, _ICat > &, const \_\_gnu\_debug::Safe\_iterator< _Ilte, _ISeq, _ICat > &, const \_\_gnu\_debug::Safe\_iterator<`  
`_Olte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI std::copy\_move\_a (const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, const`  
`\_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _Il, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > std::copy\_move\_backward\_a (_Il, _Il, const`  
`\_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _Ilte, typename _ISeq, typename _ICat, typename _Olte, typename _OSeq, typename _OCat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Olte, _OSeq, _OCat > std::copy\_move\_backward\_a (const`  
`\_\_gnu\_debug::Safe\_iterator< _Ilte, _ISeq, _ICat > &, const \_\_gnu\_debug::Safe\_iterator< _Ilte, _ISeq, _`  
`ICat > &, const \_\_gnu\_debug::Safe\_iterator< _Olte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI std::copy\_move\_backward\_a (const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &,`  
`const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<typename _Il1, typename _Il2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::equal\_aux (_Il1, _Il1, const \_\_gnu\_debug::Safe\_iterator< _Il2, _Seq2, _Cat2 > &)`
- `template<typename _Il1, typename _Seq1, typename _Cat1, typename _Il2>`  
`constexpr bool std::equal\_aux (const \_\_gnu\_debug::Safe\_iterator< _Il1, _Seq1, _Cat1 > &, const`  
`\_\_gnu\_debug::Safe\_iterator< _Il1, _Seq1, _Cat1 > &, _Il2)`
- `template<typename _Il1, typename _Seq1, typename _Cat1, typename _Il2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::equal\_aux (const \_\_gnu\_debug::Safe\_iterator< _Il1, _Seq1, _Cat1 > &, const`  
`\_\_gnu\_debug::Safe\_iterator< _Il1, _Seq1, _Cat1 > &, const \_\_gnu\_debug::Safe\_iterator< _Il2, _Seq2,`  
`_Cat2 > &)`
- `template<typename _Ite, typename _Seq, typename _Cat, typename _Tp>`  
`constexpr void std::fill\_a (const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, const \_\_gnu\_debug::Safe\_iterator<`  
`_Ite, _Seq, _Cat > &, const _Tp &)`

- `template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>`  
`constexpr ::__gnu_debug::__Safe_iterator< _Ite, _Seq, _Cat > std::__fill_n_a (const ::__gnu_debug::__Safe_iterator<`  
`_Ite, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)`
- `template<typename _I1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::__lexicographical_compare_aux (_I1 __first1, _I1 __last1, const ::__gnu_debug::__Safe_iterator<`  
`_Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::__Safe_iterator< _Ite2, _Seq2, _Cat2 > &__last2)`
- `template<typename _Ite1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool std::__lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, ↵`  
`_Cat1 > &__first1, const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, _I2 __first2, _I2`  
`__last2)`
- `template<typename _Ite1, typename _Seq1, typename _Cat1, typename _Ite2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::__lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Ite1, ↵`  
`_Seq1, _Cat1 > &__first1, const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, const`  
`::__gnu_debug::__Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::__Safe_iterator< ↵`  
`_Ite2, _Seq2, _Cat2 > &__last2)`
- `template<typename _Ite, typename _Seq>`  
`constexpr decltype(std::__niter_base(std::declval< _Ite >())) std::__niter_base (const ::__gnu_debug::__Safe_iterator<`  
`_Ite, _Seq, std::random_access_iterator_tag > &__it) noexcept(std::is_nothrow_copy_constructible< _Ite >↵`  
`::value)`

### 6.263.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.264 safe\_local\_iterator.h File Reference

### Classes

- class `__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >`

### Namespaces

- namespace `__gnu_debug`

### Macros

- `#define _GLIBCXX_DEBUG_VERIFY_OPERANDS(_Lhs, _Rhs)`

### Functions

- `template<typename _Iterator, typename _UContainer>`  
`_Iterator __gnu_debug::__unsafe (const _Safe_local_iterator< _Iterator, _UContainer > &__it)`
- `template<typename _Iterator, typename _UContainer>`  
`bool __gnu_debug::__valid_range (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const`  
`_Safe_local_iterator< _Iterator, _UContainer > &__last)`
- `template<typename _Iterator, typename _UContainer>`  
`bool __gnu_debug::__valid_range (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const`  
`_Safe_local_iterator< _Iterator, _UContainer > &__last, typename _Distance_traits< _Iterator >::__type &__↵`  
`dist_info)`

### 6.264.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.265 `safe_local_iterator.tcc` File Reference

### Namespaces

- namespace `__gnu_debug`

### Macros

- `#define _GLIBCXX_DEBUG_SAFE_LOCAL_ITERATOR_TCC`

#### 6.265.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.266 `safe_sequence.h` File Reference

### Classes

- class `__gnu_debug::_After_nth_from<_Iterator>`
- class `__gnu_debug::_Equal_to<_Type>`
- class `__gnu_debug::_Not_equal_to<_Type>`
- class `__gnu_debug::_Safe_node_sequence<_Sequence>`
- class `__gnu_debug::_Safe_sequence<_Sequence>`

### Namespaces

- namespace `__gnu_debug`

#### 6.266.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.267 `safe_sequence.tcc` File Reference

### Namespaces

- namespace `__gnu_debug`

### Macros

- `#define _GLIBCXX_DEBUG_SAFE_SEQUENCE_TCC`

#### 6.267.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.268 `safe_unordered_base.h` File Reference

### Classes

- class `__gnu_debug::_Safe_local_iterator_base`
- class `__gnu_debug::_Safe_unordered_container_base`

### Namespaces

- namespace `__gnu_debug`



### 6.268.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.269 `safe_unordered_container.h` File Reference

### Classes

- class `__gnu_debug::Safe_unordered_container<_Container>`

### Namespaces

- namespace `__gnu_debug`

### 6.269.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.270 `safe_unordered_container.tcc` File Reference

### Namespaces

- namespace `__gnu_debug`

### Macros

- `#define _GLIBCXX_DEBUG_SAFE_UNORDERED_CONTAINER_TCC`

### 6.270.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.271 `set.h` File Reference

### Classes

- class `std::__debug::set<_Key, _Compare, _Allocator>`

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc>  
__detail::__synth3way_t<_Key> std::__debug::operator<=> (const set<_Key, _Compare, _Alloc> &__lhs,  
const set<_Key, _Compare, _Alloc> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__debug::operator== (const set<_Key, _Compare, _Allocator> &__lhs, const set<_Key, _Compare,  
_Allocator> &__rhs)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require<←  
Allocator<_Allocator>>>  
std::__debug::set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits<_InputIterator  
>::value_type, less< typename iterator_traits<_InputIterator>::value_type>, _Allocator>`

- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< type-  
name iterator\_traits< _InputIterator >::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::set (initializer\_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot<_Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::set (initializer\_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _<_Compare, _Allocator >`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`void std::__debug::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`  
`noexcept(/\*conditional \*/)`

### 6.271.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.272 decimal File Reference

### Classes

- class [std::decimal::decimal128](#)
- class [std::decimal::decimal32](#)
- class [std::decimal::decimal64](#)

### Namespaces

- namespace [std](#)
- namespace [std::decimal](#)

### Macros

- `#define \_DECLARE\_DECIMAL128\_COMPOUND\_ASSIGNMENT(_Op)`
- `#define \_DECLARE\_DECIMAL32\_COMPOUND\_ASSIGNMENT(_Op)`
- `#define \_DECLARE\_DECIMAL64\_COMPOUND\_ASSIGNMENT(_Op)`
- `#define \_DECLARE\_DECIMAL\_BINARY\_OP\_WITH\_DEC(_Op, _T1, _T2, _T3)`
- `#define \_DECLARE\_DECIMAL\_BINARY\_OP\_WITH\_INT(_Op, _Tp)`
- `#define \_DECLARE\_DECIMAL\_COMPARISON(_Op, _Tp)`
- `#define \_GLIBCXX\_DECIMAL`
- `#define \_GLIBCXX\_USE\_DECIMAL`

### Functions

- double [std::decimal::decimal128\\_to\\_double](#) ([decimal128](#) \_\_d)
- float [std::decimal::decimal128\\_to\\_float](#) ([decimal128](#) \_\_d)
- long double [std::decimal::decimal128\\_to\\_long\\_double](#) ([decimal128](#) \_\_d)
- long long [std::decimal::decimal128\\_to\\_long\\_long](#) ([decimal128](#) \_\_d)
- double [std::decimal::decimal32\\_to\\_double](#) ([decimal32](#) \_\_d)
- float [std::decimal::decimal32\\_to\\_float](#) ([decimal32](#) \_\_d)
- long double [std::decimal::decimal32\\_to\\_long\\_double](#) ([decimal32](#) \_\_d)
- long long [std::decimal::decimal32\\_to\\_long\\_long](#) ([decimal32](#) \_\_d)
- double [std::decimal::decimal64\\_to\\_double](#) ([decimal64](#) \_\_d)

- float **std::decimal::decimal64\_to\_float** ([decimal64](#) \_\_d)
- long double **std::decimal::decimal64\_to\_long\_double** ([decimal64](#) \_\_d)
- long long **std::decimal::decimal64\_to\_long\_long** ([decimal64](#) \_\_d)
- double **std::decimal::decimal\_to\_double** ([decimal128](#) \_\_d)
- double **std::decimal::decimal\_to\_double** ([decimal32](#) \_\_d)
- double **std::decimal::decimal\_to\_double** ([decimal64](#) \_\_d)
- float **std::decimal::decimal\_to\_float** ([decimal128](#) \_\_d)
- float **std::decimal::decimal\_to\_float** ([decimal32](#) \_\_d)
- float **std::decimal::decimal\_to\_float** ([decimal64](#) \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** ([decimal128](#) \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** ([decimal32](#) \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** ([decimal64](#) \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** ([decimal128](#) \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** ([decimal32](#) \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** ([decimal64](#) \_\_d)
- static [decimal128](#) **std::decimal::make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static [decimal128](#) **std::decimal::make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **std::decimal::make\_decimal32** (long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **std::decimal::make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **std::decimal::make\_decimal64** (long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **std::decimal::make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- bool **std::decimal::operator!=** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **std::decimal::operator!=** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **std::decimal::operator!=** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **std::decimal::operator!=** ([decimal128](#) \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** ([decimal128](#) \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **std::decimal::operator!=** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **std::decimal::operator!=** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **std::decimal::operator!=** ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** ([decimal32](#) \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **std::decimal::operator!=** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **std::decimal::operator!=** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **std::decimal::operator!=** ([decimal64](#) \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** ([decimal64](#) \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, [decimal64](#) \_\_rhs)

- bool **std::decimal::operator!=** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator\*** (int \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long long \_\_lhs, decimal64 \_\_rhs)

- Generated by Doxygen

- [illegible]



- Generated by Doxygen

- bool **std::decimal::operator<** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, unsigned long long \_\_rhs)



- Generated by Doxygen

- bool **std::decimal::operator**> (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator**> (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator**> (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator**> (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, decimal64 \_\_rhs)

- bool **std::decimal::operator**>= (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator**>= (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned long long \_\_lhs, decimal64 \_\_rhs)

### 6.272.1 Detailed Description

This is a Standard C++ Library header.

## 6.273 debug/deque File Reference

### Classes

- class **std::\_\_debug::deque**< \_Tp, \_Allocator >

### Namespaces

- namespace **std**
- namespace **std::\_\_debug**

### Macros

- #define **\_GLIBCXX\_DEBUG\_DEQUE**

### Functions

- template<typename \_InputIterator, typename \_ValT = typename iterator\_traits<\_InputIterator>::value\_type, typename \_Allocator = allocator<\_ValT>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>  
**std::\_\_debug::deque** (\_InputIterator, \_InputIterator, \_Allocator=\_Allocator()) -> deque<\_ValT, \_Allocator >
- template<typename \_Tp, typename \_Allocator = allocator<\_Tp>, typename = \_RequireAllocator<\_Allocator>>>  
**std::\_\_debug::deque** (size\_t, \_Tp, \_Allocator=\_Allocator()) -> deque<\_Tp, \_Allocator >
- template<typename \_Tp, typename \_Alloc>  
constexpr \_\_detail::\_\_synth3way\_t<\_Tp > **std::\_\_debug::operator**<=> (const deque<\_Tp, \_Alloc > &\_\_x,  
const deque<\_Tp, \_Alloc > &\_\_y)

- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`void std::__debug::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`

### 6.273.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.274 deque File Reference

### Namespaces

- namespace `std`

### Macros

- `#define __glibcxx_want_algorithm_default_value_type`
- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_containers_ranges`
- `#define __glibcxx_want_erase_if`
- `#define __glibcxx_want_nonmember_container_access`
- `#define _GLIBCXX_DEQUE`

### Typedefs

- `template<typename _Tp>`  
`using std::pmr::deque`

### 6.274.1 Detailed Description

This is a Standard C++ Library header.

## 6.275 experimental/deque File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_DEQUE`

### Typedefs

- `template<typename _Tp>`  
`using std::experimental::fundamentals_v2::pmr::deque`

### Functions

- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void std::experimental::erase (deque< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (deque< _Tp, _Alloc > &__cont, _Predicate __pred)`

### 6.275.1 Detailed Description

This is a TS C++ Library header.

## 6.276 expected File Reference

### 6.276.1 Detailed Description

This is a Standard C++ Library header.

## 6.277 lfts\_config.h File Reference

### 6.277.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.278 ext/numeric\_traits.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Macros

- `#define __glibcxx_digits10(_Tp)`
- `#define __glibcxx_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define __glibcxx_max_digits10(_Tp)`
- `#define __glibcxx_max_exponent10(_Tp)`
- `#define __GLIBCXX_INT_N_TRAITS(T, WIDTH)`

### Typedefs

- `template<typename _Tp>`  
`using \_\_gnu\_cxx::\_\_int\_traits`

### Variables

- `template<typename _Value>`  
`const int \_\_gnu\_cxx::\_\_numeric\_traits\_floating<\_Value>::\_\_digits10`
- `template<typename _Value>`  
`const bool \_\_gnu\_cxx::\_\_numeric\_traits\_floating<\_Value>::\_\_is\_signed`
- `template<typename _Value>`  
`const int \_\_gnu\_cxx::\_\_numeric\_traits\_floating<\_Value>::\_\_max\_digits10`
- `template<typename _Value>`  
`const int \_\_gnu\_cxx::\_\_numeric\_traits\_floating<\_Value>::\_\_max\_exponent10`
- `template<typename _Value>`  
`const int \_\_gnu\_cxx::\_\_numeric\_traits\_integer<\_Value>::\_\_digits`
- `template<typename _Value>`  
`const bool \_\_gnu\_cxx::\_\_numeric\_traits\_integer<\_Value>::\_\_is\_signed`
- `template<typename _Value>`  
`const _Value \_\_gnu\_cxx::\_\_numeric\_traits\_integer<\_Value>::\_\_max`
- `template<typename _Value>`  
`const _Value \_\_gnu\_cxx::\_\_numeric\_traits\_integer<\_Value>::\_\_min`

### 6.278.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.279 propagate\_const File Reference

### Classes

- class [std::experimental::fundamentals\\_v2::propagate\\_const<\\_Tp>](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_PROPAGATE_CONST`

### Typedefs

- `template<typename _Tp>`  
using [std::experimental::\\_\\_propagate\\_const\\_elem\\_type](#)

### Functions

- `template<typename _Tp>`  
`constexpr const _Tp & std::experimental::get_underlying (const propagate\_const<\_Tp> &__pt) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp & std::experimental::get_underlying (propagate\_const<\_Tp> &__pt) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (const _Tp &__t, const propagate\_const<\_Up> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (const propagate\_const<\_Tp> &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (const propagate\_const<\_Tp> &__pt, const propagate\_const<\_Up> &__pu)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator!= (const propagate\_const<\_Tp> &__pt, nullptr_t)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator!= (nullptr_t, const propagate\_const<\_Tp> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (const _Tp &__t, const propagate\_const<\_Up> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (const propagate\_const<\_Tp> &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (const propagate\_const<\_Tp> &__pt, const propagate\_const<\_Up> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (const _Tp &__t, const propagate\_const<\_Up> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (const propagate\_const<\_Tp> &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (const propagate\_const<\_Tp> &__pt, const propagate\_const<\_Up> &__pu)`

- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (const propagate\_const< _Tp > &__pt, nullptr_t)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (nullptr_t, const propagate\_const< _Tp > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr enable\_if\_t< __is_swappable< _Tp >::value, void > std::experimental::swap (propagate\_const< _Tp > &__pt, propagate\_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)`

### 6.279.1 Detailed Description

This is a TS C++ Library header.

## 6.280 `simd` File Reference

### Macros

- `#define \_\_cpp\_lib\_experimental\_parallel\_simd`
- `#define \_GLIBCXX\_EXPERIMENTAL\_SIMD`

### 6.280.1 Detailed Description

This is a TS C++ Library header.

## 6.281 `aligned_buffer.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.281.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.282 atomicity.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Macros

- `#define _GLIBCXX_READ_MEM_BARRIER`
- `#define _GLIBCXX_WRITE_MEM_BARRIER`

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_atomic\\_add](#) (volatile `_Atomic_word` \* \_\_mem, int \_\_val)
- void [\\_\\_gnu\\_cxx::\\_\\_atomic\\_add\\_dispatch](#) (`_Atomic_word` \* \_\_mem, int \_\_val)
- void [\\_\\_gnu\\_cxx::\\_\\_atomic\\_add\\_single](#) (`_Atomic_word` \* \_\_mem, int \_\_val)
- `_Atomic_word` [\\_\\_gnu\\_cxx::\\_\\_exchange\\_and\\_add](#) (volatile `_Atomic_word` \* \_\_mem, int \_\_val)
- `_Atomic_word` [\\_\\_gnu\\_cxx::\\_\\_exchange\\_and\\_add\\_dispatch](#) (`_Atomic_word` \* \_\_mem, int \_\_val)
- `_Atomic_word` [\\_\\_gnu\\_cxx::\\_\\_exchange\\_and\\_add\\_single](#) (`_Atomic_word` \* \_\_mem, int \_\_val)
- bool [\\_\\_gnu\\_cxx::\\_\\_is\\_single\\_threaded](#) () noexcept

#### 6.282.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.283 bitmap\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_mini\\_vector<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Bitmap\\_counter<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Ffit\\_finder<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::free\\_list](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::\\_\\_detail](#)

### Macros

- `#define _BALLOC_ALIGN_BYTES`

### Enumerations

- enum { `bits_per_byte` , `bits_per_block` }

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_allocate](#) (std::size\_t \* \_\_pmap, std::size\_t \_\_pos) throw ()
- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_free](#) (std::size\_t \* \_\_pmap, std::size\_t \_\_pos) throw ()
- template<typename `_ForwardIterator`, typename `_Tp`, typename `_Compare`>  
`_ForwardIterator` [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_lower\\_bound](#) (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, const `_Tp` & \_\_val, `_Compare` \_\_comp)



- `template<typename _AddrPair>`  
`std::size_t __gnu_cxx::__detail::__num_bitmaps (_AddrPair __ap)`
- `template<typename _AddrPair>`  
`std::size_t __gnu_cxx::__detail::__num_blocks (_AddrPair __ap)`
- `std::size_t __gnu_cxx::Bit_scan_forward (std::size_t __num)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator== (const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator< _Tp2 > &) throw ()`

## Variables

- `template<typename _Tp>`  
`std::size_t __gnu_cxx::bitmap_allocator< _Tp >::__S_block_size`
- `template<typename _Tp>`  
`bitmap\_allocator< _Tp >::__BPVector::size_type __gnu_cxx::bitmap_allocator< _Tp >::__S_last_dealloc_↵`  
`index`
- `template<typename _Tp>`  
`bitmap\_allocator< _Tp >::__BPVector __gnu_cxx::bitmap_allocator< _Tp >::__S_mem_blocks`
- `template<typename _Tp>`  
`bitmap\_allocator< _Tp >::__mutex_type __gnu_cxx::bitmap_allocator< _Tp >::__S_mut`

### 6.283.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 6.283.2 Macro Definition Documentation

#### `__BALLOC_ALIGN_BYTES`

```
#define __BALLOC_ALIGN_BYTES
```

The constant in the expression below is the alignment required in bytes.

## 6.284 `cast.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::Caster](#)< [\\_ToType](#) >

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _ToType, typename _FromType>`  
`_ToType __gnu_cxx::__const_pointer_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __gnu_cxx::__const_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __gnu_cxx::__dynamic_pointer_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __gnu_cxx::__dynamic_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __gnu_cxx::__reinterpret_pointer_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __gnu_cxx::__reinterpret_pointer_cast (const _FromType &__arg)`

- `template<typename _ToType, typename _FromType>`  
`_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (const _FromType &__arg)`

### 6.284.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/pointer.h>`.

## 6.285 `codecvt_specializations.h` File Reference

### Classes

- class `std::codecvt<\_InternT, \_ExternT, encoding\_state>`
- struct `\_\_gnu\_cxx::encoding\_char\_traits<\_CharT>`
- class `\_\_gnu\_cxx::encoding\_state`
- class `std::encoding\_state`

### Namespaces

- namespace `\_\_gnu\_cxx`
- namespace `std`

### Functions

- `template<typename _Tp>`  
`size_t std::\_\_iconv\_adapter (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **↵  
__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)`

### Variables

- `template<typename _InternT, typename _ExternT>`  
`locale::id std::codecvt<\_InternT, \_ExternT, encoding\_state>::id`

### 6.285.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.286 `concurrency.h` File Reference

### Classes

- class `\_\_gnu\_cxx::\_\_scoped\_lock`

### Namespaces

- namespace `\_\_gnu\_cxx`

### Enumerations

- enum `\_Lock\_policy { \_S\_single, \_S\_mutex, \_S\_atomic }`

## Functions

- void `__gnu_cxx::__throw_concurrency_lock_error()`
- void `__gnu_cxx::__throw_concurrency_unlock_error()`

## Variables

- const `_Lock_policy` `__gnu_cxx::__default_lock_policy`

### 6.286.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.287 `debug_allocator.h` File Reference

### Classes

- class `__gnu_cxx::debug_allocator<_Alloc>`

### Namespaces

- namespace `__gnu_cxx`

### 6.287.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.288 `enc_filebuf.h` File Reference

### Classes

- class `__gnu_cxx::enc_filebuf<_CharT>`

### Namespaces

- namespace `__gnu_cxx`

### 6.288.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.289 `extptr_allocator.h` File Reference

### Classes

- class `__gnu_cxx::ExtPtr_allocator<_Tp>`

### Namespaces

- namespace `__gnu_cxx`

## Functions

- `template<typename _Tp>`  
void `__gnu_cxx::swap` (`ExtPtr_allocator<_Tp>` &`__larg`, `ExtPtr_allocator<_Tp>` &`__rarg`)

### 6.289.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

An example allocator which uses an alternative pointer type from bits/pointer.h. Supports test cases which confirm container support for alternative pointers.

## 6.290 malloc\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::malloc\\_allocator< \\_Tp >](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.290.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.291 mt\_allocator.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_common\\_pool\\_policy< \\_PoolTp, \\_Thread >](#)
- class [\\_\\_gnu\\_cxx::\\_\\_mt\\_alloc< \\_Tp, \\_Poolp >](#)
- class [\\_\\_gnu\\_cxx::\\_\\_mt\\_alloc\\_base< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::\\_\\_per\\_type\\_pool\\_policy< \\_Tp, \\_PoolTp, \\_Thread >](#)
- class [\\_\\_gnu\\_cxx::\\_\\_pool< false >](#)
- class [\\_\\_gnu\\_cxx::\\_\\_pool< true >](#)
- struct [\\_\\_gnu\\_cxx::\\_\\_pool\\_base](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Macros

- `#define \_\_thread\_default`

### Typedefs

- typedef void(\* [\\_\\_gnu\\_cxx::\\_\\_destroy\\_handler](#)) (void \*)

### Functions

- template<typename [\\_Tp](#), typename [\\_Poolp](#)>  
bool [\\_\\_gnu\\_cxx::operator==](#) (const [\\_\\_mt\\_alloc< \\_Tp, \\_Poolp >](#) &, const [\\_\\_mt\\_alloc< \\_Tp, \\_Poolp >](#) &)

### 6.291.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.292 `assoc_container.hpp` File Reference

### Classes

- class `__gnu_pbds::basic_branch`< Key, Mapped, Tag, Node\_Update, Policy\_Tl, \_Alloc >
- class `__gnu_pbds::basic_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_Tl, \_Alloc >
- class `__gnu_pbds::cc_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >
- class `__gnu_pbds::gp_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >
- class `__gnu_pbds::list_update`< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >
- class `__gnu_pbds::tree`< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >
- class `__gnu_pbds::trie`< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >

### Namespaces

- namespace `__gnu_pbds`

### Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`
- `#define PB_DS_LU_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

#### 6.292.1 Detailed Description

Contains associative containers.

## 6.293 `bin_search_tree.hpp` File Reference

### Namespaces

- namespace `__gnu_pbds`

### Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node)`
- `#define PB_DS_BIN_TREE_NAME`
- `#define PB_DS_BIN_TREE_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

### Variables

- `template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>`  
`bin_search_tree_map`< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::`node_allocator` `__gnu_pbds::detail::bin_search_tree_map`< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::`s_node_allocator`

#### 6.293.1 Detailed Description

Contains an implementation class for binary search tree.

## 6.294 bin\_search\_tree\_/node\_iterators.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_TREE_CONST_NODE_ITERATOR_CLASS_C_DEC`
- `#define PB_DS_TREE_NODE_ITERATOR_CLASS_C_DEC`

#### 6.294.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 6.295 ov\_tree\_map\_/node\_iterators.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_const\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_OV_TREE_CONST_NODE_ITERATOR_C_DEC`
- `#define PB_DS_OV_TREE_NODE_ITERATOR_C_DEC`

#### 6.295.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 6.296 point\_iterators.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference >](#)
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_TREE_CONST_IT_C_DEC`
- `#define PB_DS_TREE_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_TREE_IT_C_DEC`
- `#define PB_DS_TREE_ODIR_IT_C_DEC`

### 6.296.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 6.297 `bin_search_tree_/r_erase_fn_imps.hpp` File Reference

### 6.297.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 6.298 `pat_trie_/r_erase_fn_imps.hpp` File Reference

### 6.298.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 6.299 `bin_search_tree_/rotate_fn_imps.hpp` File Reference

### 6.299.1 Detailed Description

Contains imps for rotating nodes.

## 6.300 `pat_trie_/rotate_fn_imps.hpp` File Reference

### 6.300.1 Detailed Description

Contains imps for rotating nodes.

## 6.301 `bin_search_tree_/traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.301.1 Detailed Description

Contains an implementation for `bin_search_tree_`.

## 6.302 `branch_policy/traits.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_DEBUG_VERIFY(_Cond)`

### 6.302.1 Detailed Description

Contains an implementation class for tree-like classes.

## 6.303 ov\_tree\_map\_/traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.303.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 6.304 pat\_trie\_/traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, Mapped, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, null\\_type, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.304.1 Detailed Description

Contains an implementation class for pat\_trie\_.

## 6.305 rb\_tree\_map\_/traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.305.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 6.306 splay\_tree\_/traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)



### 6.306.1 Detailed Description

Contains an implementation for `splay_tree_`.

## 6.307 `binary_heap_.hpp` File Reference

### Classes

- class `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`

### Namespaces

- namespace `__gnu_pbds`

### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_ENTRY_CMP_DEC`
- `#define PB_DS_RESIZE_POLICY_DEC`

### Variables

- `template<typename Value_Type, typename Cmp_Fn, typename _Alloc>`  
`binary_heap< Value_Type, Cmp_Fn, _Alloc >::entry_allocator __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >::s_entry_allocator`
- `template<typename Value_Type, typename Cmp_Fn, typename _Alloc>`  
`binary_heap< Value_Type, Cmp_Fn, _Alloc >::no_throw_copies_t __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >::s_no_throw_copies_ind`
- `template<typename Value_Type, typename Cmp_Fn, typename _Alloc>`  
`binary_heap< Value_Type, Cmp_Fn, _Alloc >::value_allocator __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >::s_value_allocator`

### 6.307.1 Detailed Description

Contains an implementation class for a binary heap.

## 6.308 `binary_heap_/const_iterator.hpp` File Reference

### Classes

- class `__gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >`

### Namespaces

- namespace `__gnu_pbds`

### Macros

- `#define PB_DS_BIN_HEAP_CIT_BASE`

### 6.308.1 Detailed Description

Contains an iterator class returned by the table's `const find` and `insert` methods.

## 6.309 left\_child\_next\_sibling\_heap\_/const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_const\\_iterator\\_< Node, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_BASIC_HEAP_CIT_BASE`
- `#define PB_DS_CLASS_C_DEC`

#### 6.309.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 6.310 unordered\_iterator/const\_iterator.hpp File Reference

#### 6.310.1 Detailed Description

Contains an iterator class used for const ranging over the elements of the table.

This file is intended to be included inside a class definition, with PB\_DS\_CLASS\_C\_DEC defined to the name of the enclosing class.

## 6.311 bin\_search\_tree\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 6.311.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 6.312 binary\_heap\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 6.312.1 Detailed Description

Contains an implementation class for binary\_heap\_.

## 6.313 binomial\_heap\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 6.313.1 Detailed Description

Contains an implementation for binomial\_heap\_.

## 6.314 binomial\_heap\_base\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 6.314.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 6.315 left\_child\_next\_sibling\_heap\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 6.315.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

**6.316 `ov_tree_map_/constructors_destructor_fn_impls.hpp` File Reference****6.316.1 Detailed Description**

Contains an implementation class for `ov_tree_`.

**6.317 `pairing_heap_/constructors_destructor_fn_impls.hpp` File Reference****6.317.1 Detailed Description**

Contains an implementation class for a pairing heap.

**6.318 `pat_trie_/constructors_destructor_fn_impls.hpp` File Reference****6.318.1 Detailed Description**

Contains an implementation class for `pat_trie`.

**6.319 `rb_tree_map_/constructors_destructor_fn_impls.hpp` File Reference****6.319.1 Detailed Description**

Contains an implementation for `rb_tree_`.

**6.320 `rc_binomial_heap_/constructors_destructor_fn_impls.hpp` File Reference****6.320.1 Detailed Description**

Contains an implementation for `rc_binomial_heap_`.

**6.321 `splay_tree_/constructors_destructor_fn_impls.hpp` File Reference****6.321.1 Detailed Description**

Contains an implementation class for `splay_tree_`.

**6.322 `thin_heap_/constructors_destructor_fn_impls.hpp` File Reference****6.322.1 Detailed Description**

Contains an implementation for `thin_heap_`.

**6.323 `bin_search_tree_/debug_fn_impls.hpp` File Reference****6.323.1 Detailed Description**

Contains an implementation class for `bin_search_tree_`.

**6.324 `binary_heap_/debug_fn_impls.hpp` File Reference****6.324.1 Detailed Description**

Contains an implementation class for a `binary_heap`.

**6.325 `binomial_heap_/debug_fn_impls.hpp` File Reference****6.325.1 Detailed Description**

Contains an implementation for `binomial_heap_`.

## **6.326 binomial\_heap\_base\_/debug\_fn\_imps.hpp File Reference**

### **6.326.1 Detailed Description**

Contains an implementation class for a base of binomial heaps.

## **6.327 cc\_hash\_table\_map\_/debug\_fn\_imps.hpp File Reference**

### **6.327.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

## **6.328 gp\_hash\_table\_map\_/debug\_fn\_imps.hpp File Reference**

### **6.328.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s debug-mode functions.

## **6.329 left\_child\_next\_sibling\_heap\_/debug\_fn\_imps.hpp File Reference**

### **6.329.1 Detailed Description**

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## **6.330 list\_update\_map\_/debug\_fn\_imps.hpp File Reference**

### **6.330.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

## **6.331 ov\_tree\_map\_/debug\_fn\_imps.hpp File Reference**

### **6.331.1 Detailed Description**

Contains an implementation class for ov\_tree\_.

## **6.332 pairing\_heap\_/debug\_fn\_imps.hpp File Reference**

### **6.332.1 Detailed Description**

Contains an implementation class for a pairing heap.

## **6.333 pat\_trie\_/debug\_fn\_imps.hpp File Reference**

### **6.333.1 Detailed Description**

Contains an implementation class for pat\_trie\_.

## **6.334 rb\_tree\_map\_/debug\_fn\_imps.hpp File Reference**

### **6.334.1 Detailed Description**

Contains an implementation for rb\_tree\_.

## **6.335 rc\_binomial\_heap\_/debug\_fn\_imps.hpp File Reference**

### **6.335.1 Detailed Description**

Contains an implementation for rc\_binomial\_heap\_.

### 6.336 `splay_tree_/debug_fn_imps.hpp` File Reference

#### 6.336.1 Detailed Description

Contains an implementation class for `splay_tree_`.

### 6.337 `thin_heap_/debug_fn_imps.hpp` File Reference

#### 6.337.1 Detailed Description

Contains an implementation for `thin_heap_`.

### 6.338 `entry_cmp.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, false >::type](#)

#### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.338.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 6.339 `entry_pred.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred<\\_VTp, Pred, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred<\\_VTp, Pred, \\_Alloc, true >](#)

#### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.339.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 6.340 `bin_search_tree_/erase_fn_imps.hpp` File Reference

#### 6.340.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

### 6.341 `binary_heap_/erase_fn_imps.hpp` File Reference

#### 6.341.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 6.342 binomial\_heap\_base\_/erase\_fn\_imps.hpp File Reference

### 6.342.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 6.343 cc\_hash\_table\_map\_/erase\_fn\_imps.hpp File Reference

### 6.343.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s erase related functions.

## 6.344 gp\_hash\_table\_map\_/erase\_fn\_imps.hpp File Reference

### 6.344.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s erase related functions.

## 6.345 left\_child\_next\_sibling\_heap\_/erase\_fn\_imps.hpp File Reference

### 6.345.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 6.346 list\_update\_map\_/erase\_fn\_imps.hpp File Reference

### 6.346.1 Detailed Description

Contains implementations of lu\_map\_.

## 6.347 ov\_tree\_map\_/erase\_fn\_imps.hpp File Reference

### 6.347.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 6.348 pairing\_heap\_/erase\_fn\_imps.hpp File Reference

### 6.348.1 Detailed Description

Contains an implementation class for a pairing heap.

## 6.349 pat\_trie\_/erase\_fn\_imps.hpp File Reference

### 6.349.1 Detailed Description

Contains an implementation class for pat\_trie.

## 6.350 rb\_tree\_map\_/erase\_fn\_imps.hpp File Reference

### 6.350.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 6.351 rc\_binomial\_heap\_/erase\_fn\_imps.hpp File Reference

### 6.351.1 Detailed Description

Contains an implementation for rc\_binomial\_heap\_.

**6.352 splay\_tree\_/erase\_fn\_imps.hpp File Reference****6.352.1 Detailed Description**

Contains an implementation class for splay\_tree\_.

**6.353 thin\_heap\_/erase\_fn\_imps.hpp File Reference****6.353.1 Detailed Description**

Contains an implementation for thin\_heap\_.

**6.354 bin\_search\_tree\_/find\_fn\_imps.hpp File Reference****6.354.1 Detailed Description**

Contains an implementation class for bin\_search\_tree\_.

**6.355 binary\_heap\_/find\_fn\_imps.hpp File Reference****6.355.1 Detailed Description**

Contains an implementation class for a binary\_heap.

**6.356 binomial\_heap\_base\_/find\_fn\_imps.hpp File Reference****6.356.1 Detailed Description**

Contains an implementation class for a base of binomial heaps.

**6.357 cc\_hash\_table\_map\_/find\_fn\_imps.hpp File Reference****6.357.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s find related functions.

**6.358 gp\_hash\_table\_map\_/find\_fn\_imps.hpp File Reference****6.358.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s find related functions.

**6.359 list\_update\_map\_/find\_fn\_imps.hpp File Reference****6.359.1 Detailed Description**

Contains implementations of lu\_map\_.

**6.360 pairing\_heap\_/find\_fn\_imps.hpp File Reference****6.360.1 Detailed Description**

Contains an implementation class for a pairing heap.

**6.361 pat\_trie\_/find\_fn\_imps.hpp File Reference****6.361.1 Detailed Description**

Contains an implementation class for pat\_trie.

## 6.362 rb\_tree\_map\_/find\_fn\_imps.hpp File Reference

### 6.362.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 6.363 splay\_tree\_/find\_fn\_imps.hpp File Reference

### 6.363.1 Detailed Description

Contains an implementation class for splay\_tree\_.

## 6.364 thin\_heap\_/find\_fn\_imps.hpp File Reference

### 6.364.1 Detailed Description

Contains an implementation for thin\_heap\_.

## 6.365 bin\_search\_tree\_/info\_fn\_imps.hpp File Reference

### 6.365.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 6.366 binary\_heap\_/info\_fn\_imps.hpp File Reference

### 6.366.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 6.367 cc\_hash\_table\_map\_/info\_fn\_imps.hpp File Reference

### 6.367.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s entire container info related functions.

## 6.368 gp\_hash\_table\_map\_/info\_fn\_imps.hpp File Reference

### 6.368.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s entire container info related functions.

## 6.369 left\_child\_next\_sibling\_heap\_/info\_fn\_imps.hpp File Reference

### 6.369.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 6.370 list\_update\_map\_/info\_fn\_imps.hpp File Reference

### 6.370.1 Detailed Description

Contains implementations of lu\_map\_.

## 6.371 ov\_tree\_map\_/info\_fn\_imps.hpp File Reference

### 6.371.1 Detailed Description

Contains an implementation class for ov\_tree\_.



**6.372 pat\_trie\_/info\_fn\_imps.hpp File Reference****6.372.1 Detailed Description**

Contains an implementation class for pat\_trie.

**6.373 rb\_tree\_map\_/info\_fn\_imps.hpp File Reference****6.373.1 Detailed Description**

Contains an implementation for rb\_tree\_.

**6.374 splay\_tree\_/info\_fn\_imps.hpp File Reference****6.374.1 Detailed Description**

Contains an implementation.

**6.375 bin\_search\_tree\_/insert\_fn\_imps.hpp File Reference****6.375.1 Detailed Description**

Contains an implementation class for bin\_search\_tree\_.

**6.376 binary\_heap\_/insert\_fn\_imps.hpp File Reference****6.376.1 Detailed Description**

Contains an implementation class for a binary\_heap.

**6.377 binomial\_heap\_base\_/insert\_fn\_imps.hpp File Reference****6.377.1 Detailed Description**

Contains an implementation class for a base of binomial heaps.

**6.378 cc\_hash\_table\_map\_/insert\_fn\_imps.hpp File Reference****6.378.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s insert related functions.

**6.379 gp\_hash\_table\_map\_/insert\_fn\_imps.hpp File Reference****6.379.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s insert related functions.

**6.380 left\_child\_next\_sibling\_heap\_/insert\_fn\_imps.hpp File Reference****6.380.1 Detailed Description**

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

**6.381 list\_update\_map\_/insert\_fn\_imps.hpp File Reference****6.381.1 Detailed Description**

Contains implementations of lu\_map\_.

**6.382 `ov_tree_map_/insert_fn_imps.hpp` File Reference****6.382.1 Detailed Description**

Contains an implementation class for `ov_tree_`.

**6.383 `pairing_heap_/insert_fn_imps.hpp` File Reference****6.383.1 Detailed Description**

Contains an implementation class for a pairing heap.

**6.384 `rb_tree_map_/insert_fn_imps.hpp` File Reference****6.384.1 Detailed Description**

Contains an implementation for `rb_tree_`.

**6.385 `rc_binomial_heap_/insert_fn_imps.hpp` File Reference****6.385.1 Detailed Description**

Contains an implementation for `rc_binomial_heap_`.

**6.386 `splay_tree_/insert_fn_imps.hpp` File Reference****6.386.1 Detailed Description**

Contains an implementation class for `splay_tree_`.

**6.387 `thin_heap_/insert_fn_imps.hpp` File Reference****6.387.1 Detailed Description**

Contains an implementation for `thin_heap_`.

**6.388 `bin_search_tree_/iterators_fn_imps.hpp` File Reference****6.388.1 Detailed Description**

Contains an implementation class for `bin_search_tree_`.

**6.389 `binary_heap_/iterators_fn_imps.hpp` File Reference****6.389.1 Detailed Description**

Contains an implementation class for a `binary_heap_`.

**6.390 `cc_hash_table_map_/iterators_fn_imps.hpp` File Reference****6.390.1 Detailed Description**

Contains implementations of `cc_ht_map_`'s iterators related functions, e.g., `begin()`.

**6.391 `left_child_next_sibling_heap_/iterators_fn_imps.hpp` File Reference****6.391.1 Detailed Description**

Contains an implementation class for `left_child_next_sibling_heap_`.

### 6.392 `list_update_map_/iterators_fn_imps.hpp` File Reference

#### 6.392.1 Detailed Description

Contains implementations of `lu_map_`.

### 6.393 `ov_tree_map_/iterators_fn_imps.hpp` File Reference

#### 6.393.1 Detailed Description

Contains an implementation class for `ov_tree_`.

### 6.394 `pat_trie_/iterators_fn_imps.hpp` File Reference

#### 6.394.1 Detailed Description

Contains an implementation class for `pat_trie`.

### 6.395 `binary_heap_/point_const_iterator.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator\\_< Value\\_Type, Entry, Simple, \\_Alloc >](#)

#### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.395.1 Detailed Description

Contains an iterator class returned by the table's `const` find and insert methods.

### 6.396 `left_child_next_sibling_heap_/point_const_iterator.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator\\_< Node, \\_Alloc >](#)

#### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 6.396.1 Detailed Description

Contains an iterator class returned by the table's `const` find and insert methods.

### 6.397 `unordered_iterator/point_const_iterator.hpp` File Reference

#### 6.397.1 Detailed Description

Contains an iterator class returned by the tables' `const` find and insert methods.

- This file is intended to be included inside a class definition, with `PB_DS_CLASS_C_DEC` defined to the name of the enclosing class.

## 6.398 bin\_search\_tree\_/policy\_access\_fn\_imps.hpp File Reference

### 6.398.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 6.399 binary\_heap\_/policy\_access\_fn\_imps.hpp File Reference

### 6.399.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 6.400 cc\_hash\_table\_map\_/policy\_access\_fn\_imps.hpp File Reference

### 6.400.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s policy access functions.

## 6.401 gp\_hash\_table\_map\_/policy\_access\_fn\_imps.hpp File Reference

### 6.401.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s policy access functions.

## 6.402 left\_child\_next\_sibling\_heap\_/policy\_access\_fn\_imps.hpp File Reference

### 6.402.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 6.403 ov\_tree\_map\_/policy\_access\_fn\_imps.hpp File Reference

### 6.403.1 Detailed Description

Contains an implementation class for ov\_tree.

## 6.404 pat\_trie\_/policy\_access\_fn\_imps.hpp File Reference

### 6.404.1 Detailed Description

Contains an implementation class for pat\_trie.

## 6.405 resize\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::resize\\_policy< \\_Tp >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Variables

- template<typename \_Tp>  
const \_Tp [\\_\\_gnu\\_pbds::detail::resize\\_policy< \\_Tp >::min\\_size](#)

**6.405.1 Detailed Description**

Contains an implementation class for a `binary_heap`.

**6.406 `bin_search_tree_/split_join_fn_imps.hpp` File Reference****6.406.1 Detailed Description**

Contains an implementation class for `bin_search_tree_`.

**6.407 `binary_heap_/split_join_fn_imps.hpp` File Reference****6.407.1 Detailed Description**

Contains an implementation class for a `binary_heap`.

**6.408 `binomial_heap_base_/split_join_fn_imps.hpp` File Reference****6.408.1 Detailed Description**

Contains an implementation class for a base of binomial heaps.

**6.409 `ov_tree_map_/split_join_fn_imps.hpp` File Reference****6.409.1 Detailed Description**

Contains an implementation class for `ov_tree_`.

**6.410 `pairing_heap_/split_join_fn_imps.hpp` File Reference****6.410.1 Detailed Description**

Contains an implementation class for a pairing heap.

**6.411 `rb_tree_map_/split_join_fn_imps.hpp` File Reference****6.411.1 Detailed Description**

Contains an implementation for `rb_tree_`.

**6.412 `rc_binomial_heap_/split_join_fn_imps.hpp` File Reference****6.412.1 Detailed Description**

Contains an implementation for `rc_binomial_heap_`.

**6.413 `splay_tree_/split_join_fn_imps.hpp` File Reference****6.413.1 Detailed Description**

Contains an implementation class for `splay_tree_`.

**6.414 `thin_heap_/split_join_fn_imps.hpp` File Reference****6.414.1 Detailed Description**

Contains an implementation for `thin_heap_`.

## 6.415 `binary_heap_/trace_fn_imps.hpp` File Reference

### 6.415.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 6.416 `cc_hash_table_map_/trace_fn_imps.hpp` File Reference

### 6.416.1 Detailed Description

Contains implementations of `cc_ht_map_`'s trace-mode functions.

## 6.417 `gp_hash_table_map_/trace_fn_imps.hpp` File Reference

### 6.417.1 Detailed Description

Contains implementations of `gp_ht_map_`'s trace-mode functions.

## 6.418 `left_child_next_sibling_heap_/trace_fn_imps.hpp` File Reference

### 6.418.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 6.419 `list_update_map_/trace_fn_imps.hpp` File Reference

### 6.419.1 Detailed Description

Contains implementations of `lu_map_`.

## 6.420 `pat_trie_/trace_fn_imps.hpp` File Reference

### 6.420.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 6.421 `rc_binomial_heap_/trace_fn_imps.hpp` File Reference

### 6.421.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

## 6.422 `thin_heap_/trace_fn_imps.hpp` File Reference

### 6.422.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 6.423 `binomial_heap_.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

**Macros**

- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**

**6.423.1 Detailed Description**

Contains an implementation class for a binomial heap.

**6.424 binomial\_heap\_base.hpp File Reference****Classes**

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap\\_base< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**Macros**

- #define **PB\_DS\_ASSERT\_BASE\_NODE\_CONSISTENT**(\_Node, \_Bool)
- #define **PB\_DS\_ASSERT\_VALID\_COND**(X, \_StrictlyBinomial)
- #define **PB\_DS\_B\_HEAP\_BASE**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**

**6.424.1 Detailed Description**

Contains an implementation class for a base of binomial heaps.

**6.425 branch\_policy.hpp File Reference****Classes**

- struct [\\_\\_gnu\\_pbds::detail::branch\\_policy< Node\\_Cltr, Node\\_Itr, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::branch\\_policy< Node\\_Cltr, Node\\_Cltr, \\_Alloc >](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**6.425.1 Detailed Description**

Contains a base class for branch policies.

**6.426 null\_node\_metadata.hpp File Reference****Classes**

- struct [\\_\\_gnu\\_pbds::detail::dumnode\\_const\\_iterator< Key, Data, \\_Alloc >](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

### 6.426.1 Detailed Description

Contains an implementation class for tree-like classes.

## 6.427 cc\_ht\_map\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_CC\_HASH\_NAME**
- #define **PB\_DS\_CC\_HASH\_TRAITS\_BASE**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_GEN\_POS**
- #define **PB\_DS\_HASH\_EQ\_FN\_C\_DEC**
- #define **PB\_DS\_RANGED\_HASH\_FN\_C\_DEC**

### Variables

- template<typename Key, typename Mapped, typename Hash\_Fn, typename Eq\_Fn, typename \_Alloc, bool Store\_Hash, typename Comb\_Hash\_Fn, typename Resize\_Policy>  
[cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >::const\\_iterator](#) [\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >::s\\_const\\_end\\_it](#)
- template<typename Key, typename Mapped, typename Hash\_Fn, typename Eq\_Fn, typename \_Alloc, bool Store\_Hash, typename Comb\_Hash\_Fn, typename Resize\_Policy>  
[cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >::iterator](#) [\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >::s\\_end\\_it](#)
- template<typename Key, typename Mapped, typename Hash\_Fn, typename Eq\_Fn, typename \_Alloc, bool Store\_Hash, typename Comb\_Hash\_Fn, typename Resize\_Policy>  
[cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >::entry\\_allocator](#) [\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >::s\\_entry\\_allocator](#)
- template<typename Key, typename Mapped, typename Hash\_Fn, typename Eq\_Fn, typename \_Alloc, bool Store\_Hash, typename Comb\_Hash\_Fn, typename Resize\_Policy>  
[cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >::entry\\_pointer\\_allocator](#) [\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >::s\\_entry\\_pointer\\_allocator](#)

### 6.427.1 Detailed Description

Contains an implementation class for cc\_ht\_map\_.

## 6.428 cmp\_fn\_imps.hpp File Reference

### 6.428.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s entire container comparison related functions.



## 6.429 cond\_key\_dtor\_entry\_dealtor.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::cond\\_dealtor< Entry, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.429.1 Detailed Description

Contains a conditional key destructor, used for exception handling.

## 6.430 cc\_hash\_table\_map\_/constructor\_destructor\_fn\_imps.hpp File Reference

#### 6.430.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

## 6.431 gp\_hash\_table\_map\_/constructor\_destructor\_fn\_imps.hpp File Reference

#### 6.431.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

## 6.432 list\_update\_map\_/constructor\_destructor\_fn\_imps.hpp File Reference

## 6.433 cc\_hash\_table\_map\_/constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp File Reference

#### 6.433.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

## 6.434 gp\_hash\_table\_map\_/constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp File Reference

#### 6.434.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

## 6.435 cc\_hash\_table\_map\_/constructor\_destructor\_store\_hash\_fn\_imps.hpp File Reference

#### 6.435.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

## 6.436 gp\_hash\_table\_map\_/constructor\_destructor\_store\_hash\_fn\_imps.hpp File Reference

#### 6.436.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

**6.437 cc\_hash\_table\_map\_/debug\_no\_store\_hash\_fn\_imps.hpp File Reference****6.437.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

**6.438 gp\_hash\_table\_map\_/debug\_no\_store\_hash\_fn\_imps.hpp File Reference****6.438.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s debug-mode functions.

**6.439 cc\_hash\_table\_map\_/debug\_store\_hash\_fn\_imps.hpp File Reference****6.439.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

**6.440 gp\_hash\_table\_map\_/debug\_store\_hash\_fn\_imps.hpp File Reference****6.440.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s debug-mode functions.

**6.441 entry\_list\_fn\_imps.hpp File Reference****6.441.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s entry-list related functions.

**6.442 cc\_hash\_table\_map\_/erase\_no\_store\_hash\_fn\_imps.hpp File Reference****6.442.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s erase related functions, when the hash value is not stored.

**6.443 gp\_hash\_table\_map\_/erase\_no\_store\_hash\_fn\_imps.hpp File Reference****6.443.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s erase related functions, when the hash value is not stored.

**6.444 cc\_hash\_table\_map\_/erase\_store\_hash\_fn\_imps.hpp File Reference****6.444.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s erase related functions, when the hash value is stored.

**6.445 gp\_hash\_table\_map\_/erase\_store\_hash\_fn\_imps.hpp File Reference****6.445.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s erase related functions, when the hash value is stored.

**6.446 cc\_hash\_table\_map\_/find\_store\_hash\_fn\_imps.hpp File Reference****6.446.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s find related functions, when the hash value is stored.

**6.447 gp\_hash\_table\_map\_/find\_store\_hash\_fn\_imps.hpp File Reference****6.447.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s insert related functions, when the hash value is stored.

**6.448 cc\_hash\_table\_map\_/insert\_no\_store\_hash\_fn\_imps.hpp File Reference****6.448.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s insert related functions, when the hash value is not stored.

**6.449 gp\_hash\_table\_map\_/insert\_no\_store\_hash\_fn\_imps.hpp File Reference****6.449.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s insert related functions, when the hash value is not stored.

**6.450 cc\_hash\_table\_map\_/insert\_store\_hash\_fn\_imps.hpp File Reference****6.450.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s insert related functions, when the hash value is stored.

**6.451 gp\_hash\_table\_map\_/insert\_store\_hash\_fn\_imps.hpp File Reference****6.451.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s find related functions, when the hash value is stored.

**6.452 cc\_hash\_table\_map\_/resize\_fn\_imps.hpp File Reference****6.452.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s resize related functions.

**6.453 gp\_hash\_table\_map\_/resize\_fn\_imps.hpp File Reference****6.453.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s resize related functions.

**6.454 cc\_hash\_table\_map\_/resize\_no\_store\_hash\_fn\_imps.hpp File Reference****6.454.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s resize related functions, when the hash value is not stored.

**6.455 gp\_hash\_table\_map\_/resize\_no\_store\_hash\_fn\_imps.hpp File Reference****6.455.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s resize related functions, when the hash value is not stored.

**6.456 cc\_hash\_table\_map\_/resize\_store\_hash\_fn\_imps.hpp File Reference****6.456.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s resize related functions, when the hash value is stored.

## 6.457 gp\_hash\_table\_map\_/resize\_store\_hash\_fn\_imps.hpp File Reference

### 6.457.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s resize related functions, when the hash value is stored.

## 6.458 size\_fn\_imps.hpp File Reference

### 6.458.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s entire container size related functions.

## 6.459 cond\_dealtor.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::cond\\_dealtor< Entry, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Variables

- template<typename Entry, class \_Alloc>  
[cond\\_dealtor< Entry, \\_Alloc >::entry\\_allocator](#) [\\_\\_gnu\\_pbds::detail::cond\\_dealtor< Entry, \\_Alloc >::s\\_alloc](#)

### 6.459.1 Detailed Description

Contains a conditional deallocator.

## 6.460 container\_base\_dispatch.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, cc\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, gp\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, list\\_update\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, ov\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, pat\\_trie\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, rb\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, splay\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, cc\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, gp\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, list\\_update\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, ov\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, pat\\_trie\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, rb\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, splay\\_tree\\_tag, Policy\\_TI >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CHECK_KEY_DOES_NOT_EXIST(_Key)`
- `#define PB_DS_CHECK_KEY_EXISTS(_Key)`
- `#define PB_DS_DATA_FALSE_INDICATOR`
- `#define PB_DS_DATA_TRUE_INDICATOR`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2S(X)`
- `#define PB_DS_V2S(X)`

### 6.460.1 Detailed Description

Contains associative container dispatching.

## 6.461 `debug_map_base.hpp` File Reference

### 6.461.1 Detailed Description

Contains a debug-mode base for all maps.

## 6.462 `eq_by_less.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::eq\\_by\\_less< Key, Cmp\\_Fn >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.462.1 Detailed Description

Contains an equivalence function.

## 6.463 `hash_eq_fn.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, true >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.463.1 Detailed Description

Contains 2 equivalence functions, one employing a hash value, and one ignoring it.

## 6.464 find\_no\_store\_hash\_fn\_imps.hpp File Reference

### 6.464.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s find related functions, when the hash value is not stored.

## 6.465 gp\_ht\_map\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::gp\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Probe\\_Fn, Probe\\_Fn, Resize\\_Policy>](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_GP_HASH_NAME`
- `#define PB_DS_GP_HASH_TRAITS_BASE`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_PROBE_FN_C_DEC`

### Variables

- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>  
gp\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy>::const_iterator \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy>::s\_const\_end\_it`
- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>  
gp\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy>::iterator \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy>::s\_end\_it`
- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>  
gp\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy>::entry_allocator \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy>::s\_entry\_allocator`

### 6.465.1 Detailed Description

Contains an implementation class for general probing hash.

## 6.466 iterator\_fn\_imps.hpp File Reference

### 6.466.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s iterators related functions, e.g., begin().

## 6.467 `direct_mask_range_hashing_imp.hpp` File Reference

### 6.467.1 Detailed Description

Contains a range-hashing policy implementation

## 6.468 `direct_mod_range_hashing_imp.hpp` File Reference

### 6.468.1 Detailed Description

Contains a range-hashing policy implementation

## 6.469 `linear_probe_fn_imp.hpp` File Reference

### 6.469.1 Detailed Description

Contains a probe policy implementation

## 6.470 `mask_based_range_hashing.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Variables

- `template<typename Size_Type>`  
const [mask\\_based\\_range\\_hashing< Size\\_Type >::size\\_type](#) [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >::s\\_highest\\_bit\\_1](#)
- `template<typename Size_Type>`  
const [mask\\_based\\_range\\_hashing< Size\\_Type >::size\\_type](#) [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >::s\\_num\\_bits\\_in\\_size\\_type](#)

### 6.470.1 Detailed Description

Contains a range hashing policy base.

## 6.471 `mod_based_range_hashing.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::mod\\_based\\_range\\_hashing< Size\\_Type >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.471.1 Detailed Description

Contains a range hashing policy base.

## 6.472 probe\_fn\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::probe\\_fn\\_base< \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.472.1 Detailed Description

Contains a probe policy base.

## 6.473 quadratic\_probe\_fn\_imp.hpp File Reference

### 6.473.1 Detailed Description

Contains a probe policy implementation

## 6.474 ranged\_hash\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, true >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, true >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 6.474.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

## 6.475 ranged\_probe\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, true >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Probe\\_Fn, null\\_type, false >](#)



**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**Macros**

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

**6.475.1 Detailed Description**

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probing.

**6.476 sample\_probe\_fn.hpp File Reference****Classes**

- class [\\_\\_gnu\\_pbds::sample\\_probe\\_fn](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**6.476.1 Detailed Description**

Contains a sample probe policy.

**6.477 sample\_range\_hashing.hpp File Reference****Classes**

- class [\\_\\_gnu\\_pbds::sample\\_range\\_hashing](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**6.477.1 Detailed Description**

Contains a range hashing policy.

**6.478 sample\_ranged\_hash\_fn.hpp File Reference****Classes**

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_hash\\_fn](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**6.478.1 Detailed Description**

Contains a ranged hash policy.

## 6.479 sample\_ranged\_probe\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_probe\\_fn](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.479.1 Detailed Description

Contains a ranged probe policy.

## 6.480 left\_child\_next\_sibling\_heap\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap< Value\\_Type, Cmp\\_Fn, Node\\_Metadata, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### Variables

- `template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>  
left\_child\_next\_sibling\_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::no_throw_copies_t \_\_gnu\_↵  
\_pbds::detail::left\_child\_next\_sibling\_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::s_no_↵  
throw_copies_ind`
- `template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>  
left\_child\_next\_sibling\_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::node_allocator \_\_gnu\_↵  
pbds::detail::left\_child\_next\_sibling\_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::s_node_↵  
allocator`

#### 6.480.1 Detailed Description

Contains an implementation class for a basic heap.

## 6.481 left\_child\_next\_sibling\_heap\_/node.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_< \\_Value, \\_Metadata, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.481.1 Detailed Description

Contains an implementation struct for this type of heap's node.

## 6.482 `rb_tree_map_/node.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_node\\_< Value\\_Type, Metadata, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.482.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 6.483 `splay_tree_/node.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::splay\\_tree\\_node\\_< Value\\_Type, Metadata, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.483.1 Detailed Description

Contains an implementation struct for `splay_tree_`'s node.

## 6.484 `entry_metadata_base.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.484.1 Detailed Description

Contains an implementation for a list update map.

## 6.485 `lu_map_.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_map< Key, Mapped, Eq\\_Fn, \\_Alloc, Update\\_Policy >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_LU_NAME`
- `#define PB_DS_LU_TRAITS_BASE`

## Variables

- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>`  
[lu\\_map](#)< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::entry\_allocator [\\_\\_gnu\\_pbds::detail::lu\\_map](#)< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_entry\_allocator
- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>`  
Eq\_Fn [\\_\\_gnu\\_pbds::detail::lu\\_map](#)< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_eq\_fn
- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>`  
type\_to\_type< typename [lu\\_map](#)< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::update\_metadata > \_\_↵  
[\\_\\_gnu\\_pbds::detail::lu\\_map](#)< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_metadata\_type\_indicator
- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>`  
[null\\_type](#) [\\_\\_gnu\\_pbds::detail::lu\\_map](#)< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_null\_type
- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>`  
Update\_Policy [\\_\\_gnu\\_pbds::detail::lu\\_map](#)< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_update\_↵  
policy

### 6.485.1 Detailed Description

Contains a list update map.

## 6.486 lu\_counter\_metadata.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_metadata](#)< Size\_Type >
- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base](#)< Size\_Type >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.486.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

## 6.487 sample\_update\_policy.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::sample\\_update\\_policy](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.487.1 Detailed Description

Contains a sample policy for list update containers.

## 6.488 ov\_tree\_map\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::cond\_dtor< Size\_Type >
- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_NODE_ITERATOR_NAME`
- `#define PB_DS_OV_TREE_NAME`
- `#define PB_DS_OV_TREE_TRAITS_BASE`

## Variables

- `template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>  
ov\_tree\_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::metadata_allocator \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_metadata_alloc`
- `template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>  
ov\_tree\_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::value_allocator \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_value_alloc`

### 6.488.1 Detailed Description

Contains an implementation class for `ov_tree`.

## 6.489 `pairing_heap.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::pairing\\_heap](#)< `Value_Type`, `Cmp_Fn`, `_Alloc` >

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_P_HEAP_BASE`

### 6.489.1 Detailed Description

Contains an implementation class for a pairing heap.

## 6.490 `insert_join_fn_imps.hpp` File Reference

### 6.490.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 6.491 `pat_trie.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#)< `Key`, `Mapped`, `Node_And_It_Traits`, `_Alloc` >

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_NODE_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_PAT_TRIE_NAME`
- `#define PB_DS_PAT_TRIE_TRAITS_BASE`
- `#define PB_DS_RECURSIVE_COUNT_LEAFS(X)`

## Variables

- `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>`  
[pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::head\_allocator [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::s\_head\_allocator
- `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>`  
[pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::inode\_allocator [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::s\_inode\_allocator
- `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>`  
[pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::leaf\_allocator [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::s\_leaf\_allocator

### 6.491.1 Detailed Description

Contains an implementation class for a patricia tree.

## 6.492 pat\_trie\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Clter](#)< Node, Leaf, Head, Inode, Is\_Forward\_Iterator >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Head](#)< \_ATraits, Metadata >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode](#)< \_ATraits, Metadata >
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Iter](#)< Node, Leaf, Head, Inode, Is\_Forward\_Iterator >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Leaf](#)< \_ATraits, Metadata >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata](#)< Metadata, \_Alloc >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata](#)< null\_type, \_Alloc >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_base](#)< \_ATraits, Metadata >
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer](#)< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter](#)< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode](#)< \_ATraits, Metadata >::const\_iterator
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode](#)< \_ATraits, Metadata >::iterator
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_IT_C_DEC`
- `#define PB_DS_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_IT_C_DEC`
- `#define PB_DS_ODIR_IT_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_CONST_ITERATOR_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_ITERATOR_C_DEC`

### 6.492.1 Detailed Description

Contains the base class for a patricia tree.

## 6.493 `split_fn_imps.hpp` File Reference

### 6.493.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 6.494 `synth_access_traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::synth\\_access\\_traits< Type\\_Traits, Set, \\_ATraits >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_C_DEC`
- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_T_DEC`

## Variables

- `template<typename Type_Traits, bool Set, typename _ATraits>  
integral_constant< int, Set > \_\_gnu\_pbds::detail::synth\_access\_traits< Type\_Traits, Set, \_ATraits >::s\_  
set\_ind`

### 6.494.1 Detailed Description

Contains an implementation class for a patricia tree.

## 6.495 `update_fn_imps.hpp` File Reference

### 6.495.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 6.496 priority\_queue\_base\_dispatch.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, binary\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, binomial\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, pairing\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, rc\\_binomial\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, thin\\_heap\\_tag, null\\_type>](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`

#### 6.496.1 Detailed Description

Contains an pqiative container dispatching base.

## 6.497 rb\_tree\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_map<Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc>](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RB_TREE_BASE`
- `#define PB_DS_RB_TREE_BASE_NAME`
- `#define PB_DS_RB_TREE_NAME`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

#### 6.497.1 Detailed Description

Contains an implementation for Red Black trees.

## 6.498 rc.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::rc<\\_Node, \\_Alloc>](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)



#### 6.498.1 Detailed Description

Contains a redundant (binary counter).

### 6.499 rc\_binomial\_heap\_.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::rc\\_binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

#### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### Macros

- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_RC\_C\_DEC**

#### 6.499.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

### 6.500 cc\_hash\_max\_collision\_check\_resize\_trigger\_imp.hpp File Reference

#### 6.500.1 Detailed Description

Contains a resize trigger implementation.

### 6.501 hash\_exponential\_size\_policy\_imp.hpp File Reference

#### 6.501.1 Detailed Description

Contains a resize size policy implementation.

### 6.502 hash\_load\_check\_resize\_trigger\_imp.hpp File Reference

#### 6.502.1 Detailed Description

Contains a resize trigger implementation.

### 6.503 hash\_load\_check\_resize\_trigger\_size\_base.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, true >](#)

#### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.503.1 Detailed Description

Contains an base holding size for some resize policies.

## 6.504 hash\_prime\_size\_policy\_imp.hpp File Reference

### 6.504.1 Detailed Description

Contains a resize size policy implementation.

## 6.505 hash\_standard\_resize\_policy\_imp.hpp File Reference

### 6.505.1 Detailed Description

Contains a resize policy implementation.

## 6.506 sample\_resize\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_policy](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.506.1 Detailed Description

Contains a sample resize policy for hash tables.

## 6.507 sample\_resize\_trigger.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_trigger](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.507.1 Detailed Description

Contains a sample resize trigger policy class.

## 6.508 sample\_size\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_size\\_policy](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.508.1 Detailed Description

Contains a sample size resize-policy.

## 6.509 `splay_fn_imps.hpp` File Reference

### 6.509.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 6.510 `splay_tree_.hpp` File Reference

### Classes

- class `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`

### Namespaces

- namespace `__gnu_pbds`

### Macros

- `#define PB_DS_ASSERT_BASE_NODE_CONSISTENT(_Node)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_S_TREE_BASE`
- `#define PB_DS_S_TREE_BASE_NAME`
- `#define PB_DS_S_TREE_NAME`

### 6.510.1 Detailed Description

Contains an implementation class for splay trees.

## 6.511 `standard_policies.hpp` File Reference

### Classes

- struct `__gnu_pbds::detail::default_comb_hash_fn`
- struct `__gnu_pbds::detail::default_eq_fn< Key >`
- struct `__gnu_pbds::detail::default_hash_fn< Key >`
- struct `__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >`
- struct `__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >`
- struct `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >`
- struct `__gnu_pbds::detail::default_update_policy`

### Namespaces

- namespace `__gnu_pbds`

### Macros

- `#define __dtrie_alloc`
- `#define __dtrie_string`

### Enumerations

- enum { `default_store_hash` }

### 6.511.1 Detailed Description

Contains standard policies for containers.

### 6.511.2 Enumeration Type Documentation

#### anonymous enum

`anonymous enum`

Enumeration for default behavior of stored hash data.

## 6.512 thin\_heap.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::thin\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_AUX_NULL(X)`
- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_BASE_T_P`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### Enumerations

- enum { `num_distinct_rank_bounds` }

### Variables

- static const std::size\_t [\\_\\_gnu\\_pbds::detail::g\\_a\\_rank\\_bounds](#) [num\_distinct\_rank\_bounds]

### 6.512.1 Detailed Description

Contains an implementation class for a thin heap.

## 6.513 tree\_policy/node\_metadata\_selector.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.513.1 Detailed Description

Contains an implementation class for trees.

## 6.514 trie\_policy/node\_metadata\_selector.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.514.1 Detailed Description

Contains an implementation class for tries.

## 6.515 tree\_policy/order\_statistics\_imp.hpp File Reference

### 6.515.1 Detailed Description

Contains forward declarations for order\_statistics\_key

## 6.516 trie\_policy/order\_statistics\_imp.hpp File Reference

### 6.516.1 Detailed Description

Contains forward declarations for order\_statistics\_key

## 6.517 sample\_tree\_node\_update.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_tree\\_node\\_update< Const\\_Node\\_Iter, Node\\_Iter, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.517.1 Detailed Description

Contains a samle node update functor.

## 6.518 tree\_trace\_base.hpp File Reference

### 6.518.1 Detailed Description

Contains tree-related policies.

## 6.519 prefix\_search\_node\_update\_imp.hpp File Reference

### 6.519.1 Detailed Description

Contains an implementation of prefix\_search\_node\_update.

## 6.520 sample\_trie\_access\_traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::sample\\_trie\\_access\\_traits](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.520.1 Detailed Description

Contains a sample probe policy.

## 6.521 sample\_trie\_node\_update.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_trie\\_node\\_update](#)< Node\_Cltr, Node\_Itr, ATraits, \_Alloc >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 6.521.1 Detailed Description

Contains a samle node update functor.

## 6.522 trie\_policy\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::trie\\_policy\\_base](#)< Node\_Cltr, Node\_Itr, ATraits, \_Alloc >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 6.522.1 Detailed Description

Contains an implementation of trie\_policy\_base.

## 6.523 trie\_string\_access\_traits\_imp.hpp File Reference

#### 6.523.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

## 6.524 type\_utils.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_STATIC_ASSERT(UNIQUE, E)`

## Typedefs

- `typedef std::tr1::integral_constant< int, 0 > __gnu_pbds::detail::false_type`
- `typedef std::tr1::integral_constant< int, 1 > __gnu_pbds::detail::true_type`

### 6.524.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

## 6.525 types\_traits.hpp File Reference

### Classes

- `struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >`
- `struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >`
- `struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >`
- `struct __gnu_pbds::detail::no_throw_copies< Key, null_type >`
- `struct __gnu_pbds::detail::rebind_traits< _Alloc, T >`
- `struct __gnu_pbds::detail::select_value_type< Key, Mapped >`
- `struct __gnu_pbds::detail::select_value_type< Key, null_type >`
- `struct __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >`
- `struct __gnu_pbds::detail::stored_data< _Tv, _Th, false >`
- `struct __gnu_pbds::detail::stored_hash< _Th >`
- `struct __gnu_pbds::detail::stored_value< _Tv >`
- `struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`

### Namespaces

- namespace `__gnu_pbds`

### Variables

- `template<typename Key, typename _Alloc, bool Store_Hash>`  
`__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >::s_null_type`

### 6.525.1 Detailed Description

Contains a traits class of types used by containers.

## 6.526 iterator.hpp File Reference

### 6.526.1 Detailed Description

Contains an `iterator_class` used for ranging over the elements of the table.

This file is intended to be included inside a class definition, with `PB_DS_CLASS_C_DEC` defined to the name of the enclosing class.

## 6.527 point\_iterator.hpp File Reference

### 6.527.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.

This file is intended to be included inside a class definition, with PB\_DS\_CLASS\_C\_DEC defined to the name of the enclosing class.

## 6.528 exception.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::container\\_error](#)
- struct [\\_\\_gnu\\_pbds::insert\\_error](#)
- struct [\\_\\_gnu\\_pbds::join\\_error](#)
- struct [\\_\\_gnu\\_pbds::resize\\_error](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Functions

- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_container\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_insert\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_join\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_resize\\_error\(\)](#)

### 6.528.1 Detailed Description

Contains exception classes.

## 6.529 hash\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mask\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mod\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_exponential\\_size\\_policy< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_load\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_prime\\_size\\_policy](#)
- class [\\_\\_gnu\\_pbds::hash\\_standard\\_resize\\_policy< Size\\_Policy, Trigger\\_Policy, External\\_Size\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::linear\\_probe\\_fn< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::quadratic\\_probe\\_fn< Size\\_Type >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)



## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_SIZE_BASE_C_DEC`

## Enumerations

- `enum { num_distinct_sizes_16_bit , num_distinct_sizes_32_bit , num_distinct_sizes_64_bit , num_↵ distinct_sizes }`

## Variables

- `static const std::size_t __gnu_pbds::detail::g_a_sizes [num_distinct_sizes]`

### 6.529.1 Detailed Description

Contains hash-related policies.

## 6.530 `list_update_policy.hpp` File Reference

### Classes

- `class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >`
- `class __gnu_pbds::lu_move_to_front_policy< _Alloc >`

### Namespaces

- namespace `__gnu_pbds`

### 6.530.1 Detailed Description

Contains policies for list update containers.

## 6.531 `priority_queue.hpp` File Reference

### Classes

- `class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >`

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.531.1 Detailed Description

Contains priority\_queues.

## 6.532 tag\_and\_trait.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::associative\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_branch\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::binary\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::cc\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::container\\_tag](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits< Cntnr >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< binary\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< binomial\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< cc\\_hash\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< gp\\_hash\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< list\\_update\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< ov\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< pairing\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< pat\\_trie\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< rb\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< splay\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< thin\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::gp\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::list\\_update\\_tag](#)
- struct [\\_\\_gnu\\_pbds::null\\_node\\_update< \\_Tp1, \\_Tp2, \\_Tp3, \\_Tp4 >](#)
- struct [\\_\\_gnu\\_pbds::null\\_type](#)
- struct [\\_\\_gnu\\_pbds::ov\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pairing\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pat\\_trie\\_tag](#)
- struct [\\_\\_gnu\\_pbds::point\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::priority\\_queue\\_tag](#)
- struct [\\_\\_gnu\\_pbds::range\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::rb\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::rc\\_binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::sequence\\_tag](#)
- struct [\\_\\_gnu\\_pbds::splay\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::string\\_tag](#)
- struct [\\_\\_gnu\\_pbds::thin\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trie\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trivial\\_iterator\\_tag](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Typedefs

- typedef void [\\_\\_gnu\\_pbds::trivial\\_iterator\\_difference\\_type](#)

### 6.532.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

## 6.533 [tree\\_policy.hpp](#) File Reference

### Classes

- class [\\_\\_gnu\\_pbds::tree\\_order\\_statistics\\_node\\_update](#)< Node\_Cltr, Node\_Itr, Cmp\_Fn, \_Alloc >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_BRANCH\_POLICY\_BASE**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**

### 6.533.1 Detailed Description

Contains tree-related policies.

## 6.534 [trie\\_policy.hpp](#) File Reference

### Classes

- class [\\_\\_gnu\\_pbds::trie\\_order\\_statistics\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- class [\\_\\_gnu\\_pbds::trie\\_prefix\\_search\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- struct [\\_\\_gnu\\_pbds::trie\\_string\\_access\\_traits](#)< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_TRIE\_POLICY\_BASE**

## Variables

- `template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename ↵_Alloc>`  
`detail::integral_constant< int, Reverse > __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max↵_E_Val, Reverse, _Alloc >::s_rev_ind`

### 6.534.1 Detailed Description

Contains trie-related policies.

## 6.535 pod\_char\_traits.h File Reference

### Classes

- struct `std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >`
- struct `__gnu_cxx::character< _Value, _Int, _St >`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Functions

- `template<typename _Value, typename _Int, typename _St>`  
`bool __gnu_cxx::operator< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<typename _Value, typename _Int, typename _St>`  
`bool __gnu_cxx::operator== (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`

### 6.535.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.536 pointer.h File Reference

### Classes

- struct `__gnu_cxx::Invalid_type`
- class `__gnu_cxx::Pointer_adapter< _Storage_policy >`
- class `__gnu_cxx::Relative_pointer_impl< _Tp >`
- class `__gnu_cxx::Relative_pointer_impl< const _Tp >`
- class `__gnu_cxx::Std_pointer_impl< _Tp >`
- struct `__gnu_cxx::Unqualified_type< _Tp >`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Macros

- `#define _CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define _GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

## Functions

- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT>`  
`std::basic_ostream< _CharT, _Traits > &__gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`

- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`

### 6.536.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

## 6.537 pool\_allocator.h File Reference

### Classes

- class `__gnu_cxx::__pool_alloc< _Tp >`
- class `__gnu_cxx::__pool_alloc_base`

### Namespaces

- namespace `__gnu_cxx`

### Functions

- `template<typename _Tp>`  
`bool __gnu_cxx::operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`

### Variables

- `template<typename _Tp>`  
`_Atomic_word __gnu_cxx::__pool_alloc< _Tp >::__S_force_new`

### 6.537.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.538 rb\_tree File Reference

### Classes

- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _RB_TREE`

### 6.538.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.539 `rc_string_base.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_rc\\_string\\_base<\\_CharT, \\_Traits, \\_Alloc>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Variables

- `template<typename _CharT, typename _Traits, typename _Alloc>  
\_\_rc\_string\_base<\_CharT, \_Traits, \_Alloc>::Rep\_empty \_\_gnu\_cxx::\_\_rc\_string\_base<\_CharT, \_Traits, \_Alloc>::S\_empty\_rep`

### 6.539.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 6.540 `rope` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::rope<\\_CharT, \\_Alloc>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::\\_\\_detail](#)
- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define __GC_CONST`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOCS(__a)`
- `#define __STATIC_IF_SGI_ALLOC`
- `#define __STL_FREE_STRING(__s, __l, __a)`
- `#define __STL_ROPE_FROM_UNOWNED_CHAR_PTR(__s, __size, __a)`
- `#define __ROPE`

## Typedefs

- typedef [rope](#)< char > **\_\_gnu\_cxx::crope**
- typedef [rope](#)< wchar\_t > **\_\_gnu\_cxx::wrope**

## Enumerations

- enum { **\_S\_max\_rope\_depth** }
- enum **\_Tag** { **\_S\_leaf** , **\_S\_concat** , **\_S\_substringfn** , **\_S\_function** }

## Functions

- [crope](#)::reference **\_\_gnu\_cxx::\_\_mutable\_reference\_at** ([crope](#) &\_\_c, std::size\_t \_\_i)
- template<typename \_ForwardIterator, typename \_Allocator>  
void **\_\_gnu\_cxx::Destroy\_const** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Allocator \_\_alloc)
- template<typename \_ForwardIterator, typename \_Tp>  
void **\_\_gnu\_cxx::Destroy\_const** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::allocator](#)< \_Tp >)
- template<class \_CharT>  
void **\_\_gnu\_cxx::S\_cond\_store\_eos** (\_CharT &)
- void **\_\_gnu\_cxx::S\_cond\_store\_eos** (char &\_\_c)
- void **\_\_gnu\_cxx::S\_cond\_store\_eos** (wchar\_t &\_\_c)
- template<class \_CharT>  
\_CharT **\_\_gnu\_cxx::S\_eos** (\_CharT \*)
- template<class \_CharT>  
bool **\_\_gnu\_cxx::S\_is\_basic\_char\_type** (\_CharT \*)
- bool **\_\_gnu\_cxx::S\_is\_basic\_char\_type** (char \*)
- bool **\_\_gnu\_cxx::S\_is\_basic\_char\_type** (wchar\_t \*)
- template<class \_CharT>  
bool **\_\_gnu\_cxx::S\_is\_one\_byte\_char\_type** (\_CharT \*)
- bool **\_\_gnu\_cxx::S\_is\_one\_byte\_char\_type** (char \*)
- template<class \_CharT, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const \_Rope\_char\_ptr\_proxy< \_CharT, \_Alloc > &\_\_x, const \_Rope\_char\_ptr\_proxy< \_CharT, \_Alloc > &\_\_y)
- template<class \_CharT, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const \_Rope\_const\_iterator< \_CharT, \_Alloc > &\_\_x, const \_Rope\_const\_iterator< \_CharT, \_Alloc > &\_\_y)
- template<class \_CharT, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const \_Rope\_iterator< \_CharT, \_Alloc > &\_\_x, const \_Rope\_iterator< \_CharT, \_Alloc > &\_\_y)
- template<class \_CharT, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const [rope](#)< \_CharT, \_Alloc > &\_\_x, const [rope](#)< \_CharT, \_Alloc > &\_\_y)
- template<class \_CharT, class \_Alloc>  
\_Rope\_const\_iterator< \_CharT, \_Alloc > **\_\_gnu\_cxx::operator+** (const \_Rope\_const\_iterator< \_CharT, \_Alloc > &\_\_x, std::ptrdiff\_t \_\_n)
- template<class \_CharT, class \_Alloc>  
\_Rope\_iterator< \_CharT, \_Alloc > **\_\_gnu\_cxx::operator+** (const \_Rope\_iterator< \_CharT, \_Alloc > &\_\_x, std::ptrdiff\_t \_\_n)
- template<class \_CharT, class \_Alloc>  
[rope](#)< \_CharT, \_Alloc > **\_\_gnu\_cxx::operator+** (const [rope](#)< \_CharT, \_Alloc > &\_\_left, \_CharT \_\_right)
- template<class \_CharT, class \_Alloc>  
[rope](#)< \_CharT, \_Alloc > **\_\_gnu\_cxx::operator+** (const [rope](#)< \_CharT, \_Alloc > &\_\_left, const \_CharT \*\_\_right)
- template<class \_CharT, class \_Alloc>  
[rope](#)< \_CharT, \_Alloc > **\_\_gnu\_cxx::operator+** (const [rope](#)< \_CharT, \_Alloc > &\_\_left, const [rope](#)< \_CharT, \_Alloc > &\_\_right)



- `template<class _CharT, class _Alloc>  
_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (std::ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc>  
_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc>  
rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc>  
rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const _CharT * __right)`
- `template<class _CharT, class _Alloc>  
rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc>  
std::ptrdiff_t __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>  
_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>  
std::ptrdiff_t __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>  
_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>  
bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>  
bool __gnu_cxx::operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>  
bool __gnu_cxx::operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Traits, class _Alloc>  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `template<class _CharT, class _Alloc>  
bool __gnu_cxx::operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>  
bool __gnu_cxx::operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>  
bool __gnu_cxx::operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>  
bool __gnu_cxx::operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>  
bool __gnu_cxx::operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>  
bool __gnu_cxx::operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`void __gnu_cxx::swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<class _CharT, class _Alloc>`  
`void __gnu_cxx::swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`

## Variables

- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`
- `template<class _CharT, class _Alloc>`  
`const rope< _CharT, _Alloc >::size_type __gnu_cxx::rope< _CharT, _Alloc >::npos`

### 6.540.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.541 ropeimpl.h File Reference

### Namespaces

- namespace `__gnu_cxx`

### Macros

- `#define __GLIBCXX__`
- `#define __glibcxx_assert(cond)`
- `#define __N(msgid)`
- `#define __NO_CTYPE`
- `#define _GLIBCXX11_DEPRECATED`
- `#define _GLIBCXX11_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX11_USE_C99_MATH`
- `#define _GLIBCXX11_USE_C99_STDIO`
- `#define _GLIBCXX11_USE_C99_STDLIB`
- `#define _GLIBCXX11_USE_C99_WCHAR`
- `#define _GLIBCXX14_DEPRECATED`

- `#define _GLIBCXX14_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX17_DEPRECATED`
- `#define _GLIBCXX17_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX20_DEPRECATED`
- `#define _GLIBCXX20_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX23_DEPRECATED`
- `#define _GLIBCXX23_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX26_DEPRECATED`
- `#define _GLIBCXX26_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX98_USE_C99_COMPLEX`
- `#define _GLIBCXX98_USE_C99_MATH`
- `#define _GLIBCXX98_USE_C99_STDIO`
- `#define _GLIBCXX98_USE_C99_STDLIB`
- `#define _GLIBCXX98_USE_C99_WCHAR`
- `#define _GLIBCXX_ASSERT_FAIL(_Condition)`
- `#define _GLIBCXX_ATOMIC_WORD_BUILTINS`
- `#define _GLIBCXX_AUTO_CAST(X)`
- `#define _GLIBCXX_BEGIN_EXTERN_C`
- `#define _GLIBCXX_BEGIN_INLINE_ABI_NAMESPACE(X)`
- `#define _GLIBCXX_BEGIN_NAMESPACE_ALGO`
- `#define _GLIBCXX_BEGIN_NAMESPACE_CONTAINER`
- `#define _GLIBCXX_BEGIN_NAMESPACE_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_VERSION`
- `#define _GLIBCXX_CAN_ALIGNAS_DESTRUCTIVE_SIZE`
- `#define _GLIBCXX_CPU_DEFINES`
- `#define _GLIBCXX_CXX_CONFIG_H`
- `#define _GLIBCXX_DEFAULT_ABI_TAG`
- `#define _GLIBCXX_DEPRECATED`
- `#define _GLIBCXX_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX_DOXYGEN_ONLY(X)`
- `#define _GLIBCXX_END_EXTERN_C`
- `#define _GLIBCXX_END_INLINE_ABI_NAMESPACE(X)`
- `#define _GLIBCXX_END_NAMESPACE_ALGO`
- `#define _GLIBCXX_END_NAMESPACE_CONTAINER`
- `#define _GLIBCXX_END_NAMESPACE_CXX11`
- `#define _GLIBCXX_END_NAMESPACE_LDBL`
- `#define _GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_END_NAMESPACE_VERSION`
- `#define _GLIBCXX_EXTERN_TEMPLATE`
- `#define _GLIBCXX_EXTERN_TEMPLATE`
- `#define _GLIBCXX_FAST_MATH`
- `#define _GLIBCXX_FULLY_DYNAMIC_STRING`
- `#define _GLIBCXX_HAS_BUILTIN(B)`
- `#define _GLIBCXX_HAVE__CXA_THREAD_ATEXIT_IMPL`
- `#define _GLIBCXX_HAVE_ACOSF`
- `#define _GLIBCXX_HAVE_ACOSL`
- `#define _GLIBCXX_HAVE_ALIGNED_ALLOC`
- `#define _GLIBCXX_HAVE_ARC4RANDOM`
- `#define _GLIBCXX_HAVE_ARPA_INET_H`

- #define \_GLIBCXX\_HAVE\_AS\_SYMVER\_DIRECTIVE
- #define \_GLIBCXX\_HAVE\_ASINF
- #define \_GLIBCXX\_HAVE\_ASINL
- #define \_GLIBCXX\_HAVE\_AT\_QUICK\_EXIT
- #define \_GLIBCXX\_HAVE\_ATAN2F
- #define \_GLIBCXX\_HAVE\_ATAN2L
- #define \_GLIBCXX\_HAVE\_ATANF
- #define \_GLIBCXX\_HAVE\_ATANL
- #define \_GLIBCXX\_HAVE\_ATOMIC\_LOCK\_POLICY
- #define \_GLIBCXX\_HAVE\_ATTRIBUTE\_VISIBILITY
- #define \_GLIBCXX\_HAVE\_BUILTIN\_HAS\_UNIQ\_OBJ\_REP
- #define \_GLIBCXX\_HAVE\_BUILTIN\_IS\_AGGREGATE
- #define \_GLIBCXX\_HAVE\_BUILTIN\_LAUNDER
- #define \_GLIBCXX\_HAVE\_C99\_FLT\_EVAL\_TYPES
- #define \_GLIBCXX\_HAVE\_CEILF
- #define \_GLIBCXX\_HAVE\_CEILL
- #define \_GLIBCXX\_HAVE\_COMPLEX\_H
- #define \_GLIBCXX\_HAVE\_COSF
- #define \_GLIBCXX\_HAVE\_COSHF
- #define \_GLIBCXX\_HAVE\_COSHL
- #define \_GLIBCXX\_HAVE\_COSL
- #define \_GLIBCXX\_HAVE\_DECL\_STRNLEN
- #define \_GLIBCXX\_HAVE\_DIRENT\_H
- #define \_GLIBCXX\_HAVE\_DIRFD
- #define \_GLIBCXX\_HAVE\_DLFCN\_H
- #define \_GLIBCXX\_HAVE\_ENDIAN\_H
- #define \_GLIBCXX\_HAVE\_EXCEPTION\_PTR\_SINCE\_GCC46
- #define \_GLIBCXX\_HAVE\_EXECINFO\_H
- #define \_GLIBCXX\_HAVE\_EXPF
- #define \_GLIBCXX\_HAVE\_EXPL
- #define \_GLIBCXX\_HAVE\_FABSF
- #define \_GLIBCXX\_HAVE\_FABSL
- #define \_GLIBCXX\_HAVE\_FCNTL\_H
- #define \_GLIBCXX\_HAVE\_FDOPENDIR
- #define \_GLIBCXX\_HAVE\_FENV\_H
- #define \_GLIBCXX\_HAVE\_FINITE
- #define \_GLIBCXX\_HAVE\_FINITEF
- #define \_GLIBCXX\_HAVE\_FINITEL
- #define \_GLIBCXX\_HAVE\_FLOAT\_H
- #define \_GLIBCXX\_HAVE\_FLOORF
- #define \_GLIBCXX\_HAVE\_FLOORL
- #define \_GLIBCXX\_HAVE\_FMODF
- #define \_GLIBCXX\_HAVE\_FMODL
- #define \_GLIBCXX\_HAVE\_FREXPF
- #define \_GLIBCXX\_HAVE\_FREXPL
- #define \_GLIBCXX\_HAVE\_GETENTROPY
- #define \_GLIBCXX\_HAVE\_GETIPINFO
- #define \_GLIBCXX\_HAVE\_GETS
- #define \_GLIBCXX\_HAVE\_HYPOT
- #define \_GLIBCXX\_HAVE\_HYPOTF
- #define \_GLIBCXX\_HAVE\_HYPOTL

- `#define _GLIBCXX_HAVE_ICONV`
- `#define _GLIBCXX_HAVE_INTTYPES_H`
- `#define _GLIBCXX_HAVE_IS_CONSTANT_EVALUATED`
- `#define _GLIBCXX_HAVE_ISINF`
- `#define _GLIBCXX_HAVE_ISINFF`
- `#define _GLIBCXX_HAVE_ISINFL`
- `#define _GLIBCXX_HAVE_ISNAN`
- `#define _GLIBCXX_HAVE_ISNANF`
- `#define _GLIBCXX_HAVE_ISNANL`
- `#define _GLIBCXX_HAVE_ISWBLANK`
- `#define _GLIBCXX_HAVE_LC_MESSAGES`
- `#define _GLIBCXX_HAVE_LDEXPF`
- `#define _GLIBCXX_HAVE_LDEXPL`
- `#define _GLIBCXX_HAVE_LIBINTL_H`
- `#define _GLIBCXX_HAVE_LIMIT_AS`
- `#define _GLIBCXX_HAVE_LIMIT_DATA`
- `#define _GLIBCXX_HAVE_LIMIT_FSIZE`
- `#define _GLIBCXX_HAVE_LIMIT_RSS`
- `#define _GLIBCXX_HAVE_LIMIT_VMEM`
- `#define _GLIBCXX_HAVE_LINK`
- `#define _GLIBCXX_HAVE_LINK_H`
- `#define _GLIBCXX_HAVE_LINUX_FUTEX`
- `#define _GLIBCXX_HAVE_LINUX_RANDOM_H`
- `#define _GLIBCXX_HAVE_LINUX_TYPES_H`
- `#define _GLIBCXX_HAVE_LOCALE_H`
- `#define _GLIBCXX_HAVE_LOG10F`
- `#define _GLIBCXX_HAVE_LOG10L`
- `#define _GLIBCXX_HAVE_LOGF`
- `#define _GLIBCXX_HAVE_LOGL`
- `#define _GLIBCXX_HAVE_LSEEK`
- `#define _GLIBCXX_HAVE_MBSTATE_T`
- `#define _GLIBCXX_HAVE_MEMALIGN`
- `#define _GLIBCXX_HAVE_MEMORY_H`
- `#define _GLIBCXX_HAVE_MODFF`
- `#define _GLIBCXX_HAVE_MODFL`
- `#define _GLIBCXX_HAVE_NETDB_H`
- `#define _GLIBCXX_HAVE_NETINET_IN_H`
- `#define _GLIBCXX_HAVE_NETINET_TCP_H`
- `#define _GLIBCXX_HAVE_O_NONBLOCK`
- `#define _GLIBCXX_HAVE_OPENAT`
- `#define _GLIBCXX_HAVE_POLL`
- `#define _GLIBCXX_HAVE_POLL_H`
- `#define _GLIBCXX_HAVE_POSIX_MEMALIGN`
- `#define _GLIBCXX_HAVE_POWF`
- `#define _GLIBCXX_HAVE_POWL`
- `#define _GLIBCXX_HAVE_QUICK_EXIT`
- `#define _GLIBCXX_HAVE_READLINK`
- `#define _GLIBCXX_HAVE_S_ISREG`
- `#define _GLIBCXX_HAVE_SECURE_GETENV`
- `#define _GLIBCXX_HAVE_SETENV`
- `#define _GLIBCXX_HAVE_SINCOS`

- `#define _GLIBCXX_HAVE_SINCOSF`
- `#define _GLIBCXX_HAVE_SINCOSL`
- `#define _GLIBCXX_HAVE_SINF`
- `#define _GLIBCXX_HAVE_SINHF`
- `#define _GLIBCXX_HAVE_SINHL`
- `#define _GLIBCXX_HAVE_SINL`
- `#define _GLIBCXX_HAVE_SOCKETATMARK`
- `#define _GLIBCXX_HAVE_SQRTF`
- `#define _GLIBCXX_HAVE_SQRTL`
- `#define _GLIBCXX_HAVE_STACKTRACE`
- `#define _GLIBCXX_HAVE_STDALIGN_H`
- `#define _GLIBCXX_HAVE_STDBOOL_H`
- `#define _GLIBCXX_HAVE_STDINT_H`
- `#define _GLIBCXX_HAVE_STDLIB_H`
- `#define _GLIBCXX_HAVE_STRERROR_L`
- `#define _GLIBCXX_HAVE_STRERROR_R`
- `#define _GLIBCXX_HAVE_STRING_H`
- `#define _GLIBCXX_HAVE_STRINGS_H`
- `#define _GLIBCXX_HAVE_STRTOF`
- `#define _GLIBCXX_HAVE_STRTOLD`
- `#define _GLIBCXX_HAVE_STRUCT_DIRENT_D_TYPE`
- `#define _GLIBCXX_HAVE_STRXFRM_L`
- `#define _GLIBCXX_HAVE_SYMLINK`
- `#define _GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT`
- `#define _GLIBCXX_HAVE_SYS_IOCTL_H`
- `#define _GLIBCXX_HAVE_SYS_IPC_H`
- `#define _GLIBCXX_HAVE_SYS_MMAN_H`
- `#define _GLIBCXX_HAVE_SYS_PARAM_H`
- `#define _GLIBCXX_HAVE_SYS_PTRACE_H`
- `#define _GLIBCXX_HAVE_SYS_RESOURCE_H`
- `#define _GLIBCXX_HAVE_SYS_SEM_H`
- `#define _GLIBCXX_HAVE_SYS_SOCKET_H`
- `#define _GLIBCXX_HAVE_SYS_STAT_H`
- `#define _GLIBCXX_HAVE_SYS_STATVFS_H`
- `#define _GLIBCXX_HAVE_SYS_SYSINFO_H`
- `#define _GLIBCXX_HAVE_SYS_TIME_H`
- `#define _GLIBCXX_HAVE_SYS_TYPES_H`
- `#define _GLIBCXX_HAVE_SYS_UIO_H`
- `#define _GLIBCXX_HAVE_TANF`
- `#define _GLIBCXX_HAVE_TANHF`
- `#define _GLIBCXX_HAVE_TANHL`
- `#define _GLIBCXX_HAVE_TANL`
- `#define _GLIBCXX_HAVE_TGMATH_H`
- `#define _GLIBCXX_HAVE_TIMESPEC_GET`
- `#define _GLIBCXX_HAVE_TLS`
- `#define _GLIBCXX_HAVE_TRUNCATE`
- `#define _GLIBCXX_HAVE_UCHAR_H`
- `#define _GLIBCXX_HAVE_UNISTD_H`
- `#define _GLIBCXX_HAVE_UNLINKAT`
- `#define _GLIBCXX_HAVE_USELOCALE`
- `#define _GLIBCXX_HAVE_UTIME_H`

- #define \_GLIBCXX\_HAVE\_VFWSCANF
- #define \_GLIBCXX\_HAVE\_VSWSCANF
- #define \_GLIBCXX\_HAVE\_VWSCANF
- #define \_GLIBCXX\_HAVE\_WCHAR\_H
- #define \_GLIBCXX\_HAVE\_WCSTOF
- #define \_GLIBCXX\_HAVE\_WCTYPE\_H
- #define \_GLIBCXX\_HAVE\_WRITEV
- #define \_GLIBCXX\_HOSTED
- #define \_GLIBCXX\_ICONV\_CONST
- #define \_GLIBCXX\_INLINE\_VERSION
- #define \_GLIBCXX\_LT\_OBJDIR
- #define \_GLIBCXX\_MANGLE\_SIZE\_T
- #define \_GLIBCXX\_MAY\_HAVE\_\_\_CXA\_THREAD\_ATEXIT\_IMPL
- #define \_GLIBCXX\_NAMESPACE\_CXX11
- #define \_GLIBCXX\_NAMESPACE\_LDBL
- #define \_GLIBCXX\_NAMESPACE\_LDBL\_OR\_CXX11
- #define \_GLIBCXX\_NO\_OBSOLETE\_ISINF\_ISNAN\_DYNAMIC
- #define \_GLIBCXX\_NODISCARD
- #define \_GLIBCXX\_NOEXCEPT\_PARM
- #define \_GLIBCXX\_NOEXCEPT\_QUAL
- #define \_GLIBCXX\_OS\_DEFINES
- #define \_GLIBCXX\_PACKAGE\_\_GLIBCXX\_VERSION
- #define \_GLIBCXX\_PACKAGE\_BUGREPORT
- #define \_GLIBCXX\_PACKAGE\_NAME
- #define \_GLIBCXX\_PACKAGE\_STRING
- #define \_GLIBCXX\_PACKAGE\_TARNAME
- #define \_GLIBCXX\_PACKAGE\_URL
- #define \_GLIBCXX\_PARALLEL\_FEATURES\_H
- #define \_GLIBCXX\_RELEASE
- #define \_GLIBCXX\_RES\_LIMITS
- #define \_GLIBCXX\_STATIC\_TZDATA
- #define \_GLIBCXX\_STD\_A
- #define \_GLIBCXX\_STD\_C
- #define \_GLIBCXX\_STDC\_HEADERS
- #define \_GLIBCXX\_STDIO\_EOF
- #define \_GLIBCXX\_STDIO\_SEEK\_CUR
- #define \_GLIBCXX\_STDIO\_SEEK\_END
- #define \_GLIBCXX\_SYMVER
- #define \_GLIBCXX\_SYMVER\_GNU
- #define \_GLIBCXX\_TXN\_SAFE
- #define \_GLIBCXX\_TXN\_SAFE\_DYN
- #define \_GLIBCXX\_USE\_ALLOCATOR\_NEW
- #define \_GLIBCXX\_USE\_BUILTIN\_TRAIT(BT)
- #define \_GLIBCXX\_USE\_C11\_UCHAR\_CXX11
- #define \_GLIBCXX\_USE\_C99
- #define \_GLIBCXX\_USE\_C99\_COMPLEX\_TR1
- #define \_GLIBCXX\_USE\_C99\_CTYPE
- #define \_GLIBCXX\_USE\_C99\_CTYPE\_TR1
- #define \_GLIBCXX\_USE\_C99\_FENV
- #define \_GLIBCXX\_USE\_C99\_FENV\_TR1
- #define \_GLIBCXX\_USE\_C99\_INTTYPES

- #define \_GLIBCXX\_USE\_C99\_INTPYPES\_TR1
- #define \_GLIBCXX\_USE\_C99\_INTPYPES\_WCHAR\_T
- #define \_GLIBCXX\_USE\_C99\_INTPYPES\_WCHAR\_T\_TR1
- #define \_GLIBCXX\_USE\_C99\_MATH\_FUNCS
- #define \_GLIBCXX\_USE\_C99\_MATH\_TR1
- #define \_GLIBCXX\_USE\_C99\_STDINT
- #define \_GLIBCXX\_USE\_C99\_STDINT\_TR1
- #define \_GLIBCXX\_USE\_CHDIR
- #define \_GLIBCXX\_USE\_CHMOD
- #define \_GLIBCXX\_USE\_CLOCK\_MONOTONIC
- #define \_GLIBCXX\_USE\_CLOCK\_REALTIME
- #define \_GLIBCXX\_USE\_DECIMAL\_FLOAT
- #define \_GLIBCXX\_USE\_DEV\_RANDOM
- #define \_GLIBCXX\_USE\_DUAL\_ABI
- #define \_GLIBCXX\_USE\_FCHMOD
- #define \_GLIBCXX\_USE\_FCHMODAT
- #define \_GLIBCXX\_USE\_FSEEKO\_FTELLO
- #define \_GLIBCXX\_USE\_GET\_NPROCS
- #define \_GLIBCXX\_USE\_GETCWD
- #define \_GLIBCXX\_USE\_GETTIMEOFDAY
- #define \_GLIBCXX\_USE\_INIT\_PRIORITY\_ATTRIBUTE
- #define \_GLIBCXX\_USE\_LFS
- #define \_GLIBCXX\_USE\_LONG\_LONG
- #define \_GLIBCXX\_USE\_LSTAT
- #define \_GLIBCXX\_USE\_MKDIR
- #define \_GLIBCXX\_USE\_NANOSLEEP
- #define \_GLIBCXX\_USE\_NL\_LANGINFO\_L
- #define \_GLIBCXX\_USE\_NLS
- #define \_GLIBCXX\_USE\_PROC\_SELF\_STATUS
- #define \_GLIBCXX\_USE\_PTHREAD\_COND\_CLOCKWAIT
- #define \_GLIBCXX\_USE\_PTHREAD\_MUTEX\_CLOCKLOCK
- #define \_GLIBCXX\_USE\_PTHREAD\_RWLOCK\_CLOCKLOCK
- #define \_GLIBCXX\_USE\_RANDOM\_TR1
- #define \_GLIBCXX\_USE\_REALPATH
- #define \_GLIBCXX\_USE\_SC\_NPROCESSORS\_ONLN
- #define \_GLIBCXX\_USE\_SCHED\_YIELD
- #define \_GLIBCXX\_USE\_SENDFILE
- #define \_GLIBCXX\_USE\_ST\_MTIM
- #define \_GLIBCXX\_USE\_STD\_SPEC\_FUNCS
- #define \_GLIBCXX\_USE\_STRUCT\_TM\_TM\_ZONE
- #define \_GLIBCXX\_USE\_TMPNAM
- #define \_GLIBCXX\_USE\_UCHAR\_C8RTOMB\_MBRTOC8\_CXX20
- #define \_GLIBCXX\_USE\_UCHAR\_C8RTOMB\_MBRTOC8\_FCHAR8\_T
- #define \_GLIBCXX\_USE\_UTILITY
- #define \_GLIBCXX\_USE\_UTILITY
- #define \_GLIBCXX\_USE\_WCHAR\_T
- #define \_GLIBCXX\_VERBOSE
- #define \_GLIBCXX\_VERBOSE\_ASSERT
- #define \_GLIBCXX\_VISIBILITY(V)
- #define \_GLIBCXX\_X86\_RDRAND
- #define \_GLIBCXX\_X86\_RDSEED
- #define \_GLIBCXX\_ZONEINFO\_DIR
- #define \_GTHREAD\_USE\_MUTEX\_TIMEDLOCK
- #define \_REQUIRES\_FREESTANDING\_H



## Functions

- `template<class _CharT, class _Traits>`  
`void __gnu_cxx::Rope_fill (std::basic_ostream< _CharT, _Traits > &__o, std::size_t __n)`
- `template<class _CharT>`  
`bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator>`  
`void __gnu_cxx::Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc>`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`

## Variables

- `template<class _CharT, class _Alloc>`  
`_CharT __gnu_cxx::rope< _CharT, _Alloc >::_S_empty_c_str[1]`
- `template<class _CharT, class _Alloc>`  
`const unsigned long __gnu_cxx::rope< _CharT, _Alloc >::_S_min_len [int(__detail::_S_max_rope_depth)+1]`

### 6.541.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/rope>`.

## 6.542 slist File Reference

### Classes

- class `__gnu_cxx::slist< _Tp, _Alloc >`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Macros

- `#define _SLIST`

### Functions

- `_Slist_node_base * __gnu_cxx::__slist_make_link (_Slist_node_base * __prev_node, _Slist_node_base * __new_node)`
- `_Slist_node_base * __gnu_cxx::__slist_previous (_Slist_node_base * __head, const _Slist_node_base * __node)`
- `const _Slist_node_base * __gnu_cxx::__slist_previous (const _Slist_node_base * __head, const _Slist_node_base * __node)`
- `_Slist_node_base * __gnu_cxx::__slist_reverse (_Slist_node_base * __node)`
- `std::size_t __gnu_cxx::__slist_size (_Slist_node_base * __node)`

- void `__gnu_cxx::slist_splice_after` (`_Slist_node_base *__pos`, `_Slist_node_base *__before_first`, `_Slist_node_base *__before_last`)
- void `__gnu_cxx::slist_splice_after` (`_Slist_node_base *__pos`, `_Slist_node_base *__head`)
- template<class `_Tp`, class `_Alloc`>  
bool `__gnu_cxx::operator!=` (const `slist<_Tp, _Alloc>` &`SL1`, const `slist<_Tp, _Alloc>` &`SL2`)
- template<class `_Tp`, class `_Alloc`>  
bool `__gnu_cxx::operator<` (const `slist<_Tp, _Alloc>` &`SL1`, const `slist<_Tp, _Alloc>` &`SL2`)
- template<class `_Tp`, class `_Alloc`>  
bool `__gnu_cxx::operator<=` (const `slist<_Tp, _Alloc>` &`SL1`, const `slist<_Tp, _Alloc>` &`SL2`)
- template<class `_Tp`, class `_Alloc`>  
bool `__gnu_cxx::operator==` (const `slist<_Tp, _Alloc>` &`SL1`, const `slist<_Tp, _Alloc>` &`SL2`)
- template<class `_Tp`, class `_Alloc`>  
bool `__gnu_cxx::operator>` (const `slist<_Tp, _Alloc>` &`SL1`, const `slist<_Tp, _Alloc>` &`SL2`)
- template<class `_Tp`, class `_Alloc`>  
bool `__gnu_cxx::operator>=` (const `slist<_Tp, _Alloc>` &`SL1`, const `slist<_Tp, _Alloc>` &`SL2`)
- template<class `_Tp`, class `_Alloc`>  
void `__gnu_cxx::swap` (`slist<_Tp, _Alloc>` &`x`, `slist<_Tp, _Alloc>` &`y`)

#### 6.542.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.543 sso\_string\_base.h File Reference

### Namespaces

- namespace `__gnu_cxx`

#### 6.543.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 6.544 stdio\_filebuf.h File Reference

### Classes

- class `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`

### Namespaces

- namespace `__gnu_cxx`

#### 6.544.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.545 stdio\_sync\_filebuf.h File Reference

### Classes

- class `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`

### Namespaces

- namespace `__gnu_cxx`

### 6.545.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.546 `string_conversions.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>  
_Ret \_\_gnu\_cxx::\_\_stoa (_TRet>(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const  
_CharT *__str, std::size_t *__idx, _Base... __base)`
- `template<typename _String, typename _CharT = typename _String::value_type>  
_String \_\_gnu\_cxx::\_\_to\_xstring (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT *__fmt,...)`

### 6.546.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.547 `throw_allocator.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::limit\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::annotate\\_base](#)
- struct [\\_\\_gnu\\_cxx::condition\\_base](#)
- struct [\\_\\_gnu\\_cxx::forced\\_error](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::group\\_adjustor](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_limit >](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_random >](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::limit\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::never\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::never\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition](#)
- class [\\_\\_gnu\\_cxx::throw\\_allocator\\_base< \\_Tp, \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_limit< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_random< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_base< \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_random](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Functions

- `void __gnu_cxx::throw_forced_error ()`
- `template<typename _Cond>  
throw_value_base< _Cond > __gnu_cxx::operator* (const throw_value_base< _Cond > &__a, const  
throw_value_base< _Cond > &__b)`
- `template<typename _Cond>  
throw_value_base< _Cond > __gnu_cxx::operator+ (const throw_value_base< _Cond > &__a, const  
throw_value_base< _Cond > &__b)`
- `template<typename _Cond>  
throw_value_base< _Cond > __gnu_cxx::operator- (const throw_value_base< _Cond > &__a, const  
throw_value_base< _Cond > &__b)`
- `template<typename _Cond>  
bool __gnu_cxx::operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond >  
&__b)`
- `std::ostream & __gnu_cxx::operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _Tp, typename _Cond>  
bool __gnu_cxx::operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp,  
_Cond > &)`
- `template<typename _Cond>  
bool __gnu_cxx::operator== (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond >  
&__b)`
- `template<typename _Cond>  
void __gnu_cxx::swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`

### 6.547.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (`throw_value`, `throw_allocator`) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type `forced_exception_error`.

## 6.548 `type_traits.h` File Reference

### Namespaces

- namespace `__gnu_cxx`

### Functions

- `template<typename _Type>  
constexpr bool __gnu_cxx::__is_null_pointer (_Type *__ptr)`
- `template<typename _Type>  
constexpr bool __gnu_cxx::__is_null_pointer (_Type)`
- `constexpr bool __gnu_cxx::__is_null_pointer (std::nullptr_t)`

### 6.548.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.549 `typelist.h` File Reference

### Namespaces

- namespace `__gnu_cxx`
- namespace `__gnu_cxx::typelist`

## Macros

- `#define _GLIBCXX_TYPELIST_CHAIN1(X0)`
- `#define _GLIBCXX_TYPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`
- `#define _GLIBCXX_TYPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`
- `#define _GLIBCXX_TYPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- `#define _GLIBCXX_TYPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`
- `#define _GLIBCXX_TYPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TYPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TYPELIST_CHAIN16(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)`
- `#define _GLIBCXX_TYPELIST_CHAIN17(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)`
- `#define _GLIBCXX_TYPELIST_CHAIN18(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)`
- `#define _GLIBCXX_TYPELIST_CHAIN19(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)`
- `#define _GLIBCXX_TYPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TYPELIST_CHAIN20(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)`
- `#define _GLIBCXX_TYPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TYPELIST_CHAIN4(X0, X1, X2, X3)`
- `#define _GLIBCXX_TYPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TYPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TYPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TYPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TYPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

## Functions

- `template<typename Fn, typename Typelist>`  
`void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Fn, typename Typelist>`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV>`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Gn, typename Typelist>`  
`void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV>`  
`void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`

### 6.549.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

## 6.550 `vstring.h` File Reference

### Classes

- class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`
- struct `std::hash<__gnu_cxx::__u16vstring>`
- struct `std::hash<__gnu_cxx::__u32vstring>`
- struct `std::hash<__gnu_cxx::__vstring>`
- struct `std::hash<__gnu_cxx::__wvstring>`

## Namespaces

- namespace `__gnu_cxx`
- namespace `std`

## Functions

- [illegible]



- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string<  
_CharT, _Traits, _Alloc, _Base > & __str)`
- `double __gnu_cxx::stod (const __vstring & __str, std::size_t * __idx=0)`
- `double __gnu_cxx::stod (const __wvstring & __str, std::size_t * __idx=0)`
- `float __gnu_cxx::stof (const __vstring & __str, std::size_t * __idx=0)`
- `float __gnu_cxx::stof (const __wvstring & __str, std::size_t * __idx=0)`
- `int __gnu_cxx::stoi (const __vstring & __str, std::size_t * __idx=0, int __base=10)`
- `int __gnu_cxx::stoi (const __wvstring & __str, std::size_t * __idx=0, int __base=10)`
- `long __gnu_cxx::stol (const __vstring & __str, std::size_t * __idx=0, int __base=10)`
- `long __gnu_cxx::stol (const __wvstring & __str, std::size_t * __idx=0, int __base=10)`
- `long double __gnu_cxx::stold (const __vstring & __str, std::size_t * __idx=0)`
- `long double __gnu_cxx::stold (const __wvstring & __str, std::size_t * __idx=0)`
- `long long __gnu_cxx::stoll (const __vstring & __str, std::size_t * __idx=0, int __base=10)`
- `long long __gnu_cxx::stoll (const __wvstring & __str, std::size_t * __idx=0, int __base=10)`
- `unsigned long __gnu_cxx::stoul (const __vstring & __str, std::size_t * __idx=0, int __base=10)`
- `unsigned long __gnu_cxx::stoul (const __wvstring & __str, std::size_t * __idx=0, int __base=10)`
- `unsigned long long __gnu_cxx::stoull (const __vstring & __str, std::size_t * __idx, int __base=10)`
- `unsigned long long __gnu_cxx::stoull (const __wvstring & __str, std::size_t * __idx=0, int __base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
void __gnu_cxx::swap ( __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, __versa_string< _CharT, _↵  
_Traits, _Alloc, _Base > & __rhs)`
- `__vstring __gnu_cxx::to_string (double __val)`
- `__vstring __gnu_cxx::to_string (float __val)`
- `__vstring __gnu_cxx::to_string (int __val)`
- `__vstring __gnu_cxx::to_string (long __val)`
- `__vstring __gnu_cxx::to_string (long double __val)`
- `__vstring __gnu_cxx::to_string (long long __val)`
- `__vstring __gnu_cxx::to_string (unsigned __val)`
- `__vstring __gnu_cxx::to_string (unsigned long __val)`
- `__vstring __gnu_cxx::to_string (unsigned long long __val)`
- `__wvstring __gnu_cxx::to_wstring (double __val)`
- `__wvstring __gnu_cxx::to_wstring (float __val)`
- `__wvstring __gnu_cxx::to_wstring (int __val)`
- `__wvstring __gnu_cxx::to_wstring (long __val)`
- `__wvstring __gnu_cxx::to_wstring (long double __val)`
- `__wvstring __gnu_cxx::to_wstring (long long __val)`
- `__wvstring __gnu_cxx::to_wstring (unsigned __val)`
- `__wvstring __gnu_cxx::to_wstring (unsigned long __val)`
- `__wvstring __gnu_cxx::to_wstring (unsigned long long __val)`

### 6.550.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.551 vstring.tcc File Reference

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`



## Macros

- `#define _VSTRING_TCC`

## Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string<  
_CharT, _Traits, _Alloc, _Base > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( _CharT __lhs, const __versa_string<  
_CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT * __lhs, const  
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string<  
_CharT, _Traits, _Alloc, _Base > & __str)`

## Variables

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _↵  
_Traits, _Alloc, _Base >::npos`

### 6.551.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 6.552 vstring\_fwd.h File Reference

### Namespaces

- namespace `__gnu_cxx`

### Typedefs

- `typedef __versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base > __gnu_↵  
cxx::__rc_string`
- `typedef __vstring __gnu_cxx::__sso_string`
- `typedef __versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base  
> __gnu_cxx::__u16rc_string`
- `typedef __u16vstring __gnu_cxx::__u16sso_string`
- `typedef __versa_string< char16_t > __gnu_cxx::__u16vstring`

- typedef `__versa_string`< `char32_t`, `std::char_traits`< `char32_t`>, `std::allocator`< `char32_t`>, `__rc_string_base`> `__gnu_cxx::__u32rc_string`
- typedef `__u32vstring` `__gnu_cxx::__u32sso_string`
- typedef `__versa_string`< `char32_t`> `__gnu_cxx::__u32vstring`
- typedef `__versa_string`< `char`> `__gnu_cxx::__vstring`
- typedef `__versa_string`< `wchar_t`, `std::char_traits`< `wchar_t`>, `std::allocator`< `wchar_t`>, `__rc_string_base`> `__gnu_cxx::__wrc_string`
- typedef `__wvstring` `__gnu_cxx::__wsso_string`
- typedef `__versa_string`< `wchar_t`> `__gnu_cxx::__wvstring`

### 6.552.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 6.553 `vstring_util.h` File Reference

### Namespaces

- namespace `__gnu_cxx`

### 6.553.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 6.554 `fenv.h` File Reference

### 6.554.1 Detailed Description

This is a Standard C++ Library header.

## 6.555 `experimental/filesystem` File Reference

### Macros

- `#define __cpp_lib_experimental_filesystem`
- `#define _GLIBCXX_EXPERIMENTAL_FILESYSTEM`

### 6.555.1 Detailed Description

This is a TS C++ Library header.

## 6.556 `filesystem` File Reference

### Macros

- `#define __glibcxx_want_filesystem`
- `#define _GLIBCXX_FILESYSTEM`

### 6.556.1 Detailed Description

This is a Standard C++ Library header.

## 6.557 flat\_map File Reference

### Macros

- `#define __glibcxx_want_flat_map`
- `#define _GLIBCXX_FLAT_MAP`

### 6.557.1 Detailed Description

This is a Standard C++ Library header.

## 6.558 flat\_set File Reference

### Macros

- `#define __glibcxx_want_flat_set`
- `#define _GLIBCXX_FLAT_SET`

### 6.558.1 Detailed Description

This is a Standard C++ Library header.

## 6.559 format File Reference

### Macros

- `#define __glibcxx_want_format`
- `#define __glibcxx_want_format_ranges`
- `#define __glibcxx_want_format_uchar`
- `#define _GLIBCXX_FORMAT`

### 6.559.1 Detailed Description

This is a Standard C++ Library header.

## 6.560 debug/forward\_list File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_forward\\_list< \\_SafeSequence >](#)
- class [std::\\_\\_debug::forward\\_list< \\_Tp, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Macros

- `#define __glibcxx_check_valid_fl_range(_First, _Last, _Dist)`
- `#define _GLIBCXX20_ONLY(__expr)`
- `#define _GLIBCXX_DEBUG_FORWARD_LIST`
- `#define _GLIBCXX_FWDLIST_REMOVE_RETURN_TYPE_TAG`

## Functions

- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::forward_list (size_t, _Tp, _Allocator=_Allocator()) -> forward_list< _Tp, _Allocator >`
- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp > std::__debug::operator<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc>`  
`void std::__debug::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`

### 6.560.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.561 experimental/forward\_list File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_FORWARD_LIST`

### Typedefs

- `template<typename _Tp>`  
`using std::experimental::fundamentals_v2::pmr::forward_list`

## Functions

- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void std::experimental::erase (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)`

### 6.561.1 Detailed Description

This is a TS C++ Library header.

## 6.562 forward\_list File Reference

### Namespaces

- namespace `std`

## Macros

- `#define __glibcxx_want_algorithm_default_value_type`
- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_containers_ranges`
- `#define __glibcxx_want_erase_if`
- `#define __glibcxx_want_incomplete_container_elements`
- `#define __glibcxx_want_list_remove_return_type`
- `#define __glibcxx_want_nonmember_container_access`
- `#define _GLIBCXX_FORWARD_LIST`

## Typedefs

- `template<typename _Tp>`  
`using std::pmr::forward_list`

### 6.562.1 Detailed Description

This is a Standard C++ Library header.

## 6.563 fstream File Reference

### Classes

- class [std::basic\\_filebuf< \\_CharT, \\_Traits >](#)
- class [std::basic\\_fstream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_ifstream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_ofstream< \\_CharT, \\_Traits >](#)

### Namespaces

- namespace [std](#)

## Macros

- `#define __glibcxx_want_fstream_native_handle`
- `#define _GLIBCXX_BUFSIZ`
- `#define _GLIBCXX_FSTREAM`

## Typedefs

- `template<typename _Path, typename _Result = _Path, typename _Path2 = decltype(std::declval<_Path&>().make_preferred().filename())>`  
`using std::_If_fs_path`

## Functions

- `template<class _CharT, class _Traits>`  
`void std::swap (basic\_filebuf< \_CharT, \_Traits > &__x, basic\_filebuf< \_CharT, \_Traits > &__y)`
- `template<class _CharT, class _Traits>`  
`void std::swap (basic\_fstream< \_CharT, \_Traits > &__x, basic\_fstream< \_CharT, \_Traits > &__y)`
- `template<class _CharT, class _Traits>`  
`void std::swap (basic\_ifstream< \_CharT, \_Traits > &__x, basic\_ifstream< \_CharT, \_Traits > &__y)`
- `template<class _CharT, class _Traits>`  
`void std::swap (basic\_ofstream< \_CharT, \_Traits > &__x, basic\_ofstream< \_CharT, \_Traits > &__y)`

### 6.563.1 Detailed Description

This is a Standard C++ Library header.

## 6.564 experimental/functional File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_boyer_moore_searching`
- `#define __cpp_lib_experimental_not_fn`
- `#define _GLIBCXX_EXPERIMENTAL_FUNCTIONAL`

### Typedefs

- `template<typename _RAIter, typename _Hash, typename _Pred, typename _Val = typename iterator_traits<_RAIter>::value_type, typename _Diff = typename iterator_traits<_RAIter>::difference_type>`  
using [std::experimental::\\_\\_boyer\\_moore\\_base\\_t](#)

### Functions

- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal_to<>>`  
`boyer_moore_horspool_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::make\_boyer\_moore\_horspool\_searcher`  
`(_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal_to<>>`  
`boyer_moore_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::make\_boyer\_moore\_searcher`  
`(_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>`  
`default_searcher<_ForwardIterator, _BinaryPredicate> std::experimental::make\_default\_searcher (_ForwardIterator __pat_first, _ForwardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _Fn>`  
`auto std::experimental::not\_fn (_Fn && __fn) noexcept(std::is\_nothrow\_constructible< std::decay\_t< _Fn >, _Fn && >::value)`

### Variables

- `template<typename _Tp>`  
`constexpr bool std::experimental::is\_bind\_expression\_v`
- `template<typename _Tp>`  
`constexpr int std::experimental::is\_placeholder\_v`

### 6.564.1 Detailed Description

This is a TS C++ Library header.

## 6.565 ext/functional File Reference

### Classes

- class [\\_\\_gnu\\_cxx::binary\\_compose< \\_Operation1, \\_Operation2, \\_Operation3 >](#)
- struct [\\_\\_gnu\\_cxx::constant\\_binary\\_fun< \\_Result, \\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::constant\\_unary\\_fun< \\_Result, \\_Argument >](#)
- struct [\\_\\_gnu\\_cxx::constant\\_void\\_fun< \\_Result >](#)
- struct [\\_\\_gnu\\_cxx::project1st< \\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::project2nd< \\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::select1st< \\_Pair >](#)
- struct [\\_\\_gnu\\_cxx::select2nd< \\_Pair >](#)
- class [\\_\\_gnu\\_cxx::subtractive\\_rng](#)
- class [\\_\\_gnu\\_cxx::unary\\_compose< \\_Operation1, \\_Operation2 >](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Macros

- `#define \_EXT\_FUNCTIONAL`

### Functions

- [template<class \\_Operation1, class \\_Operation2> unary\\_compose< \\_Operation1, \\_Operation2 > \\_\\_gnu\\_cxx::compose1](#) (const \_Operation1 &\_\_fn1, const \_↵  
Operation2 &\_\_fn2)
- [template<class \\_Operation1, class \\_Operation2, class \\_Operation3> binary\\_compose< \\_Operation1, \\_Operation2, \\_Operation3 > \\_\\_gnu\\_cxx::compose2](#) (const \_Operation1 &\_\_fn1,  
const \_Operation2 &\_\_fn2, const \_Operation3 &\_\_fn3)
- [template<class \\_Result> constant\\_void\\_fun< \\_Result > \\_\\_gnu\\_cxx::constant0](#) (const \_Result &\_\_val)
- [template<class \\_Result> constant\\_unary\\_fun< \\_Result, \\_Result > \\_\\_gnu\\_cxx::constant1](#) (const \_Result &\_\_val)
- [template<class \\_Result> constant\\_binary\\_fun< \\_Result, \\_Result, \\_Result > \\_\\_gnu\\_cxx::constant2](#) (const \_Result &\_\_val)
- [template<class \\_Tp> \\_Tp \\_\\_gnu\\_cxx::identity\\_element](#) (std::multiplies< \_Tp >)
- [template<class \\_Tp> \\_Tp \\_\\_gnu\\_cxx::identity\\_element](#) (std::plus< \_Tp >)
- [template<class \\_Ret, class \\_Tp, class \\_Arg> std::const\\_mem\\_fun1\\_t< \\_Ret, \\_Tp, \\_Arg > \\_\\_gnu\\_cxx::mem\\_fun1](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg) const)
- [template<class \\_Ret, class \\_Tp, class \\_Arg> std::mem\\_fun1\\_t< \\_Ret, \\_Tp, \\_Arg > \\_\\_gnu\\_cxx::mem\\_fun1](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg))
- [template<class \\_Ret, class \\_Tp, class \\_Arg> std::const\\_mem\\_fun1\\_ref\\_t< \\_Ret, \\_Tp, \\_Arg > \\_\\_gnu\\_cxx::mem\\_fun1\\_ref](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg) const)
- [template<class \\_Ret, class \\_Tp, class \\_Arg> std::mem\\_fun1\\_ref\\_t< \\_Ret, \\_Tp, \\_Arg > \\_\\_gnu\\_cxx::mem\\_fun1\\_ref](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg))

#### 6.565.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.566 functional File Reference

### Classes

- class `std::_Not_fn<_Fn>`
- struct `std::_Placeholder<_Num>`
- struct `std::is_bind_expression<_Tp>`
- struct `std::is_bind_expression<_Bind<_Signature>>`
- struct `std::is_bind_expression<_Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<const _Bind<_Signature>>`
- struct `std::is_bind_expression<const _Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<const volatile _Bind<_Signature>>`
- struct `std::is_bind_expression<const volatile _Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<volatile _Bind<_Signature>>`
- struct `std::is_bind_expression<volatile _Bind_result<_Result, _Signature>>`
- struct `std::is_placeholder<_Tp>`
- struct `std::is_placeholder<_Placeholder<_Num>>`

### Namespaces

- namespace `std`
- namespace `std::placeholders`

### Macros

- `#define __glibcxx_want_bind_back`
- `#define __glibcxx_want_bind_front`
- `#define __glibcxx_want_boyer_moore_searcher`
- `#define __glibcxx_want_constexpr_functional`
- `#define __glibcxx_want_copyable_function`
- `#define __glibcxx_want_function_ref`
- `#define __glibcxx_want_invoke`
- `#define __glibcxx_want_invoke_r`
- `#define __glibcxx_want_move_only_function`
- `#define __glibcxx_want_not_fn`
- `#define __glibcxx_want_ranges`
- `#define __glibcxx_want_reference_wrapper`
- `#define __glibcxx_want_transparent_operators`
- `#define _GLIBCXX_FUNCTIONAL`
- `#define _GLIBCXX_NOT_FN_CALL_OP(_QUALS)`
- `#define _GLIBCXX_PLACEHOLDER`

### Typedefs

- `template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>`  
`using std::__is_socketlike`



## Functions

- `template<typename _Func, typename... _BoundArgs>`  
`constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type std::bind (_Func &&←  
__f, _BoundArgs &&... __args)`
- `template<typename _Result, typename _Func, typename... _BoundArgs>`  
`constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type std::bind (_Func &&__f, _BoundArgs &&...  
__args)`
- `template<typename _Tp, typename _Class>`  
`constexpr _Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::*__pm) noexcept`

## Variables

- `const _Placeholder< 1 > std::placeholders::_1`
- `const _Placeholder< 10 > std::placeholders::_10`
- `const _Placeholder< 11 > std::placeholders::_11`
- `const _Placeholder< 12 > std::placeholders::_12`
- `const _Placeholder< 13 > std::placeholders::_13`
- `const _Placeholder< 14 > std::placeholders::_14`
- `const _Placeholder< 15 > std::placeholders::_15`
- `const _Placeholder< 16 > std::placeholders::_16`
- `const _Placeholder< 17 > std::placeholders::_17`
- `const _Placeholder< 18 > std::placeholders::_18`
- `const _Placeholder< 19 > std::placeholders::_19`
- `const _Placeholder< 2 > std::placeholders::_2`
- `const _Placeholder< 20 > std::placeholders::_20`
- `const _Placeholder< 21 > std::placeholders::_21`
- `const _Placeholder< 22 > std::placeholders::_22`
- `const _Placeholder< 23 > std::placeholders::_23`
- `const _Placeholder< 24 > std::placeholders::_24`
- `const _Placeholder< 25 > std::placeholders::_25`
- `const _Placeholder< 26 > std::placeholders::_26`
- `const _Placeholder< 27 > std::placeholders::_27`
- `const _Placeholder< 28 > std::placeholders::_28`
- `const _Placeholder< 29 > std::placeholders::_29`
- `const _Placeholder< 3 > std::placeholders::_3`
- `const _Placeholder< 4 > std::placeholders::_4`
- `const _Placeholder< 5 > std::placeholders::_5`
- `const _Placeholder< 6 > std::placeholders::_6`
- `const _Placeholder< 7 > std::placeholders::_7`
- `const _Placeholder< 8 > std::placeholders::_8`
- `const _Placeholder< 9 > std::placeholders::_9`
- `template<typename _Tp>`  
`constexpr bool std::is_bind_expression_v`
- `template<typename _Tp>`  
`constexpr int std::is_placeholder_v`

### 6.566.1 Detailed Description

This is a Standard C++ Library header.

## 6.567 future File Reference

### Classes

- class `std::__basic_future< _Res >`
- class `std::future< _Res >`
- class `std::future< _Res & >`
- class `std::future< void >`
- class `std::future_error`
- struct `std::is_error_code_enum< future_errc >`
- class `std::packaged_task< _Res(_ArgTypes...)>`
- class `std::promise< _Res >`
- class `std::promise< _Res & >`
- class `std::promise< void >`
- class `std::shared_future< _Res >`
- class `std::shared_future< _Res & >`
- class `std::shared_future< void >`

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_FUTURE`

### Enumerations

- enum class `std::future_errc` { `future_already_retrieved` , `promise_already_satisfied` , `no_state` , `broken_promise` }
- enum class `std::future_status` { `ready` , `timeout` , `deferred` }
- enum class `std::launch` { `async` , `deferred` }

### Functions

- template<typename \_Fn, typename... \_Args>  
`future< __async_result_of< _Fn, _Args... > > std::async (_Fn &&__fn, _Args &&... __args)`
- template<typename \_Fn, typename... \_Args>  
`future< __async_result_of< _Fn, _Args... > > std::async (launch __policy, _Fn &&__fn, _Args &&... __args)`
- const `error_category & std::future_category ()` noexcept
- `error_code std::make_error_code (future_errc __errc)` noexcept
- `error_condition std::make_error_condition (future_errc __errc)` noexcept
- constexpr `launch std::operator& (launch __x, launch __y)` noexcept
- constexpr `launch & std::operator&= (launch &__x, launch __y)` noexcept
- constexpr `launch std::operator^ (launch __x, launch __y)` noexcept
- constexpr `launch & std::operator^= (launch &__x, launch __y)` noexcept
- constexpr `launch std::operator| (launch __x, launch __y)` noexcept
- constexpr `launch & std::operator|= (launch &__x, launch __y)` noexcept
- constexpr `launch std::operator~ (launch __x)` noexcept
- template<typename \_Fun, typename \_Signature = \_\_function\_guide\_t<\_Fun, decltype(&\_Fun::operator())>>  
`std::packaged_task (_Fun) -> packaged_task< _Signature >`
- template<typename \_Res, typename... \_ArgTypes>  
`std::packaged_task (_Res(*)(_ArgTypes...)) -> packaged_task< _Res(_ArgTypes...)>`

- `template<typename _Res, typename... _ArgTypes>`  
`void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept`
- `template<typename _Res>`  
`void std::swap (promise< _Res > &__x, promise< _Res > &__y) noexcept`

#### 6.567.1 Detailed Description

This is a Standard C++ Library header.

### 6.568 generator File Reference

#### 6.568.1 Detailed Description

This is a Standard C++ Library header.

### 6.569 inplace\_vector File Reference

#### Macros

- `#define __glibcxx_want_inplace_vector`
- `#define _GLIBCXX_INPLACE_VECTOR`

#### 6.569.1 Detailed Description

This is a Standard C++ Library header.

### 6.570 iomanip File Reference

#### Namespaces

- namespace [std](#)

#### Macros

- `#define __glibcxx_want_quoted_string_io`
- `#define _GLIBCXX_IOMANIP`

#### Functions

- `template<typename _MoneyT>`  
`_Get_money< _MoneyT > std::get\_money (_MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT>`  
`_Get_time< _CharT > std::get\_time (std::tm *__tmb, const _CharT *__fmt)`
- `template<typename _CharT, typename _Traits, typename _MoneyT>`  
`basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _Put_↵  
money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _Put_time<↵  
_CharT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _↵  
Resetiosflags __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setbase↵  
__f)`

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setfill<`  
`_CharT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setprecision`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setw`  
`__f)`
- `template ostream & std::operator<< (ostream &, _Resetiosflags)`
- `template ostream & std::operator<< (ostream &, _Setbase)`
- `template ostream & std::operator<< (ostream &, _Setfill< char >)`
- `template ostream & std::operator<< (ostream &, _Setiosflags)`
- `template ostream & std::operator<< (ostream &, _Setprecision)`
- `template ostream & std::operator<< (ostream &, _Setw)`
- `template wostream & std::operator<< (wostream &, _Resetiosflags)`
- `template wostream & std::operator<< (wostream &, _Setbase)`
- `template wostream & std::operator<< (wostream &, _Setfill< wchar_t >)`
- `template wostream & std::operator<< (wostream &, _Setiosflags)`
- `template wostream & std::operator<< (wostream &, _Setprecision)`
- `template wostream & std::operator<< (wostream &, _Setw)`
- `template<typename _CharT, typename _Traits, typename _MoneyT>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Get_money<`  
`_MoneyT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Get_time<`  
`_CharT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setbase`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setfill<`  
`_CharT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setprecision`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setw`  
`__f)`
- `template istream & std::operator>> (istream &, _Resetiosflags)`
- `template istream & std::operator>> (istream &, _Setbase)`
- `template istream & std::operator>> (istream &, _Setfill< char >)`
- `template istream & std::operator>> (istream &, _Setiosflags)`
- `template istream & std::operator>> (istream &, _Setprecision)`
- `template istream & std::operator>> (istream &, _Setw)`
- `template wistream & std::operator>> (wistream &, _Resetiosflags)`

- template [wistream](#) & [std::operator>>](#) ([wistream](#) &, [\\_Setbase](#))
- template [wistream](#) & [std::operator>>](#) ([wistream](#) &, [\\_Setfill< wchar\\_t >](#))
- template [wistream](#) & [std::operator>>](#) ([wistream](#) &, [\\_Setiosflags](#))
- template [wistream](#) & [std::operator>>](#) ([wistream](#) &, [\\_Setprecision](#))
- template [wistream](#) & [std::operator>>](#) ([wistream](#) &, [\\_Setw](#))
- template<typename [\\_MoneyT](#)>  
  [\\_Put\\_money](#)< [\\_MoneyT](#) > [std::put\\_money](#) (const [\\_MoneyT](#) &\_\_mon, bool \_\_intl=false)
- template<typename [\\_CharT](#)>  
  [\\_Put\\_time](#)< [\\_CharT](#) > [std::put\\_time](#) (const [std::tm](#) \* \_\_tmb, const [\\_CharT](#) \* \_\_fmt)
- [\\_Resetiosflags](#) [std::resetiosflags](#) ([ios\\_base::fmtflags](#) \_\_mask)
- [\\_Setbase](#) [std::setbase](#) (int \_\_base)
- template<typename [\\_CharT](#)>  
  [\\_Setfill](#)< [\\_CharT](#) > [std::setfill](#) ([\\_CharT](#) \_\_c)
- [\\_Setiosflags](#) [std::setiosflags](#) ([ios\\_base::fmtflags](#) \_\_mask)
- [\\_Setprecision](#) [std::setprecision](#) (int \_\_n)
- [\\_Setw](#) [std::setw](#) (int \_\_n)

### 6.570.1 Detailed Description

This is a Standard C++ Library header.

## 6.571 ios File Reference

### Macros

- #define [\\_\\_glibcxx\\_want\\_ios\\_noreplace](#)
- #define [\\_GLIBCXX\\_IOS](#)

### 6.571.1 Detailed Description

This is a Standard C++ Library header.

## 6.572 iosfwd File Reference

### Namespaces

- namespace [std](#)

### Macros

- #define [\\_GLIBCXX\\_IOSFWD](#)

### Typedefs

- typedef [basic\\_filebuf](#)< char > [std::filebuf](#)
- typedef [basic\\_fstream](#)< char > [std::fstream](#)
- typedef [basic\\_ifstream](#)< char > [std::ifstream](#)
- typedef [basic\\_ios](#)< char > [std::ios](#)
- typedef [basic\\_iostream](#)< char > [std::iostream](#)
- typedef [basic\\_istream](#)< char > [std::istream](#)
- typedef [basic\\_istreamstream](#)< char > [std::istreamstream](#)
- typedef [basic\\_ofstream](#)< char > [std::ofstream](#)
- typedef [basic\\_ostream](#)< char > [std::ostream](#)
- typedef [basic\\_ostreamstream](#)< char > [std::ostreamstream](#)

- typedef `basic_streambuf`< char > `std::streambuf`
- typedef `basic_stringbuf`< char > `std::stringbuf`
- typedef `basic_stringstream`< char > `std::stringstream`
- typedef `basic_filebuf`< wchar\_t > `std::wfilebuf`
- typedef `basic_fstream`< wchar\_t > `std::wfstream`
- typedef `basic_ifstream`< wchar\_t > `std::wifstream`
- typedef `basic_ios`< wchar\_t > `std::wios`
- typedef `basic_iostream`< wchar\_t > `std::wiostream`
- typedef `basic_istream`< wchar\_t > `std::wistream`
- typedef `basic_istreamstream`< wchar\_t > `std::wistreamstream`
- typedef `basic_ofstream`< wchar\_t > `std::wofstream`
- typedef `basic_ostream`< wchar\_t > `std::wostream`
- typedef `basic_ostreamstream`< wchar\_t > `std::wostreamstream`
- typedef `basic_streambuf`< wchar\_t > `std::wstreambuf`
- typedef `basic_stringbuf`< wchar\_t > `std::wstringbuf`
- typedef `basic_stringstream`< wchar\_t > `std::wstringstream`

### 6.572.1 Detailed Description

This is a Standard C++ Library header.

## 6.573 iostream File Reference

### Namespaces

- namespace `std`

### Macros

- #define `_GLIBCXX_IOSTREAM`

### Variables

- static `ios_base::Init` `std::__ioinit`

### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html> and the *I/O forward declarations*

They are required by default to cooperate with the global C library's *FILE* streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- `ostream` `std::cerr`
- `istream` `std::cin`
- `ostream` `std::clog`
- `ostream` `std::cout`
- `wostream` `std::wcerr`
- `wistream` `std::wcin`
- `wostream` `std::wclog`
- `wostream` `std::wcout`

### 6.573.1 Detailed Description

This is a Standard C++ Library header.

## 6.574 istream File Reference

### Classes

- class [std::basic\\_iostream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_istream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_istream< \\_CharT, \\_Traits >::sentry](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define \_GLIBCXX\_ISTREAM`

### Typedefs

- `template<typename _Is, typename _Tp>`  
using [std::\\_\\_rvalue\\_stream\\_extraction\\_t](#)

### Functions

- `template<typename _CharT, typename _Traits>`  
void [std::\\_\\_istream\\_extract](#) ([basic\\_istream< \\_CharT, \\_Traits >](#) &, [\\_CharT](#) \*, [streamsize](#))
- void [std::\\_\\_istream\\_extract](#) ([istream](#) &, [char](#) \*, [streamsize](#))
- `template<typename _Istream, typename _Tp>`  
[\\_\\_rvalue\\_stream\\_extraction\\_t< \\_Istream, \\_Tp >](#) [std::operator>>](#) ([\\_Istream](#) &&\_\_is, [\\_Tp](#) &&\_\_x)
- `template<typename _CharT, typename _Traits>`  
[basic\\_istream< \\_CharT, \\_Traits >](#) & [std::ws](#) ([basic\\_istream< \\_CharT, \\_Traits >](#) &\_\_is)
  
- `template<typename _CharT, typename _Traits>`  
[basic\\_istream< \\_CharT, \\_Traits >](#) & [std::operator>>](#) ([basic\\_istream< \\_CharT, \\_Traits >](#) &\_\_in, [\\_CharT](#) &\_\_c)
- `template<class _Traits>`  
[basic\\_istream< char, \\_Traits >](#) & [std::operator>>](#) ([basic\\_istream< char, \\_Traits >](#) &\_\_in, signed [char](#) &\_\_c)
- `template<class _Traits>`  
[basic\\_istream< char, \\_Traits >](#) & [std::operator>>](#) ([basic\\_istream< char, \\_Traits >](#) &\_\_in, unsigned [char](#) &\_\_c)
  
- `template<typename _CharT, typename _Traits, size_t _Num>`  
[basic\\_istream< \\_CharT, \\_Traits >](#) & [std::operator>>](#) ([basic\\_istream< \\_CharT, \\_Traits >](#) &\_\_in, [\\_CharT](#) (&\_\_s)[[\\_Num](#)])
- `template<class _Traits, size_t _Num>`  
[basic\\_istream< char, \\_Traits >](#) & [std::operator>>](#) ([basic\\_istream< char, \\_Traits >](#) &\_\_in, signed [char](#) (&\_\_s)[[\\_Num](#)])
- `template<class _Traits, size_t _Num>`  
[basic\\_istream< char, \\_Traits >](#) & [std::operator>>](#) ([basic\\_istream< char, \\_Traits >](#) &\_\_in, unsigned [char](#) (&\_\_s)[[\\_Num](#)])

#### 6.574.1 Detailed Description

This is a Standard C++ Library header.

## 6.575 experimental/iterator File Reference

### Classes

- class [std::experimental::fundamentals\\_v2::ostream\\_joiner<\\_DelimT, \\_CharT, \\_Traits>](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_ostream_joiner`
- `#define _GLIBCXX_EXPERIMENTAL_ITERATOR`

### Functions

- `template<typename _CharT, typename _Traits, typename _DelimT>  
ostream_joiner<decay_t<_DelimT>, _CharT, _Traits> std::experimental::make_ostream_joiner (basic_ostream<_CharT, _Traits> &__os, _DelimT &&__delimiter)`

#### 6.575.1 Detailed Description

This is a TS C++ Library header.

## 6.576 ext/iterator File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Macros

- `#define _EXT_ITERATOR`

### Functions

- `template<typename _InputIterator, typename _Distance>  
void __gnu_cxx::__distance (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance>  
void __gnu_cxx::__distance (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Distance>  
void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`

#### 6.576.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.577 iterator File Reference

### Macros

- `#define __glibcxx_want_array_constexpr`
- `#define __glibcxx_want_constexpr_iterator`



- `#define __glibcxx_want_make_reverse_iterator`
- `#define __glibcxx_want_move_iterator_concept`
- `#define __glibcxx_want_nonmember_container_access`
- `#define __glibcxx_want_null_iterators`
- `#define __glibcxx_want_ranges`
- `#define __glibcxx_want_ssize`
- `#define _GLIBCXX_ITERATOR`

#### 6.577.1 Detailed Description

This is a Standard C++ Library header.

### 6.578 latch File Reference

#### Macros

- `#define __glibcxx_want_latch`
- `#define _GLIBCXX_LATCH`

#### 6.578.1 Detailed Description

This is a Standard C++ Library header.

### 6.579 limits File Reference

#### Classes

- struct [std::\\_\\_numeric\\_limits\\_base](#)
- struct [std::numeric\\_limits< \\_Tp >](#)
- struct [std::numeric\\_limits< bool >](#)
- struct [std::numeric\\_limits< char >](#)
- struct [std::numeric\\_limits< char16\\_t >](#)
- struct [std::numeric\\_limits< char32\\_t >](#)
- struct [std::numeric\\_limits< double >](#)
- struct [std::numeric\\_limits< float >](#)
- struct [std::numeric\\_limits< int >](#)
- struct [std::numeric\\_limits< long >](#)
- struct [std::numeric\\_limits< long double >](#)
- struct [std::numeric\\_limits< long long >](#)
- struct [std::numeric\\_limits< short >](#)
- struct [std::numeric\\_limits< signed char >](#)
- struct [std::numeric\\_limits< unsigned char >](#)
- struct [std::numeric\\_limits< unsigned int >](#)
- struct [std::numeric\\_limits< unsigned long >](#)
- struct [std::numeric\\_limits< unsigned long long >](#)
- struct [std::numeric\\_limits< unsigned short >](#)
- struct [std::numeric\\_limits< wchar\\_t >](#)

#### Namespaces

- namespace [std](#)

## Macros

- `#define __glibcxx_concat3(P, M, S)`
- `#define __glibcxx_concat3_(P, M, S)`
- `#define __glibcxx_digits(T)`
- `#define __glibcxx_digits10(T)`
- `#define __glibcxx_digits10_b(T, B)`
- `#define __glibcxx_digits_b(T, B)`
- `#define __glibcxx_double_has_denorm_loss`
- `#define __glibcxx_double_tinyness_before`
- `#define __glibcxx_double_traps`
- `#define __glibcxx_float_has_denorm_loss`
- `#define __glibcxx_float_n(BITSIZE)`
- `#define __glibcxx_float_tinyness_before`
- `#define __glibcxx_float_traps`
- `#define __glibcxx_integral_traps`
- `#define __glibcxx_long_double_has_denorm_loss`
- `#define __glibcxx_long_double_tinyness_before`
- `#define __glibcxx_long_double_traps`
- `#define __glibcxx_max(T)`
- `#define __glibcxx_max_b(T, B)`
- `#define __glibcxx_max_digits10(T)`
- `#define __glibcxx_min(T)`
- `#define __glibcxx_min_b(T, B)`
- `#define __glibcxx_signed(T)`
- `#define __glibcxx_signed_b(T, B)`
- `#define __INT_N(TYPE, BITSIZE, EXT, UEXT)`
- `#define __INT_N_201103(TYPE)`
- `#define __INT_N_U201103(TYPE)`
- `#define __max_digits10`
- `#define _GLIBCXX_NUMERIC_LIMITS`

## Enumerations

- enum `std::float_denorm_style` { `std::denorm_indeterminate` , `std::denorm_absent` , `std::denorm_present` }
- enum `std::float_round_style` { `std::round_indeterminate` , `std::round_toward_zero` , `std::round_to_nearest` , `std::round_toward_infinity` , `std::round_toward_neg_infinity` }

### 6.579.1 Detailed Description

This is a Standard C++ Library header.

## 6.580 debug/list File Reference

### Classes

- class `std::__debug::list< _Tp, _Allocator >`

### Namespaces

- namespace `__gnu_debug`
- namespace `std`
- namespace `std::__debug`

## Macros

- `#define _GLIBCXX20_ONLY(__expr)`
- `#define _GLIBCXX_DEBUG_LIST`
- `#define _GLIBCXX_LIST_REMOVE_RETURN_TYPE_TAG`

## Functions

- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::list (size_t, _Tp, _Allocator=_Allocator()) -> list< _Tp, _Allocator >`
- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp > std::__debug::operator<=> (const list< _Tp, _Alloc > &__x, const`  
`list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`void std::__debug::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`

### 6.580.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.581 experimental/list File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_LIST`

### Typedefs

- `template<typename _Tp>`  
`using std::experimental::fundamentals_v2::pmr::list`

### Functions

- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void std::experimental::erase (list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (list< _Tp, _Alloc > &__cont, _Predicate __pred)`

### 6.581.1 Detailed Description

This is a TS C++ Library header.

## 6.582 list File Reference

### Namespaces

- namespace `std`

## Macros

- `#define __glibcxx_want_algorithm_default_value_type`
- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_containers_ranges`
- `#define __glibcxx_want_erase_if`
- `#define __glibcxx_want_incomplete_container_elements`
- `#define __glibcxx_want_list_remove_return_type`
- `#define __glibcxx_want_nonmember_container_access`
- `#define _GLIBCXX_LIST`

## Typedefs

- `template<typename _Tp>`  
`using std::pmr::list`

### 6.582.1 Detailed Description

This is a Standard C++ Library header.

## 6.583 locale File Reference

### Macros

- `#define _GLIBCXX_LOCALE`

### 6.583.1 Detailed Description

This is a Standard C++ Library header.

## 6.584 debug/map File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Macros

- `#define _GLIBCXX_DEBUG_MAP`

### 6.584.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.585 experimental/map File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_MAP`

## Typedefs

- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::map`
- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::multimap`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>  
void std::experimental::erase_if (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>  
void std::experimental::erase_if (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`

### 6.585.1 Detailed Description

This is a TS C++ Library header.

## 6.586 map File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_containers_ranges`
- `#define __glibcxx_want_erase_if`
- `#define __glibcxx_want_generic_associative_lookup`
- `#define __glibcxx_want_map_try_emplace`
- `#define __glibcxx_want_node_extract`
- `#define __glibcxx_want_nonmember_container_access`
- `#define __glibcxx_want_tuple_like`
- `#define __GLIBCXX_MAP`

## Typedefs

- `template<typename _Key, typename _Tp, typename _Cmp = std::less<_Key>>  
using std::pmr::map`
- `template<typename _Key, typename _Tp, typename _Cmp = std::less<_Key>>  
using std::pmr::multimap`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>  
map< _Key, _Tp, _Compare, _Alloc >::size_type std::erase_if (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>  
multimap< _Key, _Tp, _Compare, _Alloc >::size_type std::erase_if (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`

### 6.586.1 Detailed Description

This is a Standard C++ Library header.

## 6.587 math.h File Reference

### Functions

- `template<typename _Tp>`  
`_Tp abs (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > acos (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > acosh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > asin (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > asinh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > atan (const std::complex< _Tp > &)`
- `constexpr float atan2 (float __y, float __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > atanh (const std::complex< _Tp > &)`
- `constexpr float cbrt (float __x)`
- `constexpr float ceil (float __x)`
- `constexpr float copysign (float __x, float __y)`
- `template<typename _Tp>`  
`complex< _Tp > cos (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > cosh (const complex< _Tp > &)`
- `constexpr float erf (float __x)`
- `constexpr float erfc (float __x)`
- `template<typename _Tp>`  
`complex< _Tp > exp (const complex< _Tp > &)`
- `constexpr float exp2 (float __x)`
- `constexpr float expm1 (float __x)`
- `template<typename _Tp>`  
`_Tp fabs (const std::complex< _Tp > & __z)`
- `constexpr float fdim (float __x, float __y)`
- `constexpr float floor (float __x)`
- `constexpr float fma (float __x, float __y, float __z)`
- `constexpr float fmax (float __x, float __y)`
- `constexpr float fmin (float __x, float __y)`
- `constexpr float fmod (float __x, float __y)`
- `constexpr int fpclassify (float __x)`
- `float frexp (float __x, int * __exp)`
- `constexpr float hypot (float __x, float __y)`
- `constexpr int ilogb (float __x)`
- `constexpr bool isfinite (float __x)`
- `constexpr bool isgreater (float __x, float __y)`
- `constexpr bool isgreaterequal (float __x, float __y)`
- `constexpr bool isinf (float __x)`
- `constexpr bool isless (float __x, float __y)`
- `constexpr bool islessequal (float __x, float __y)`
- `constexpr bool islessgreater (float __x, float __y)`
- `constexpr bool isnan (float __x)`

- constexpr bool **isnormal** (float \_\_x)
- constexpr bool **isunordered** (float \_\_x, float \_\_y)
- constexpr float **ldexp** (float \_\_x, int \_\_exp)
- constexpr float **lgamma** (float \_\_x)
- constexpr long long **llrint** (float \_\_x)
- constexpr long long **llround** (float \_\_x)
- template<typename \_Tp>  
complex< \_Tp > **log** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **log10** (const complex< \_Tp > &)
- constexpr float **log1p** (float \_\_x)
- constexpr float **log2** (float \_\_x)
- constexpr float **logb** (float \_\_x)
- constexpr long **lrint** (float \_\_x)
- constexpr long **lround** (float \_\_x)
- float **modf** (float \_\_x, float \*\_\_iptr)
- constexpr float **nearbyint** (float \_\_x)
- constexpr float **nextafter** (float \_\_x, float \_\_y)
- constexpr float **nexttoward** (float \_\_x, long double \_\_y)
- template<typename \_Tp>  
complex< \_Tp > **pow** (const complex< \_Tp > &, int)
- constexpr float **remainder** (float \_\_x, float \_\_y)
- float **remquo** (float \_\_x, float \_\_y, int \*\_\_pquo)
- constexpr float **rint** (float \_\_x)
- constexpr float **round** (float \_\_x)
- constexpr float **scalbln** (float \_\_x, long \_\_ex)
- constexpr float **scalbn** (float \_\_x, int \_\_ex)
- constexpr bool **signbit** (float \_\_x)
- template<typename \_Tp>  
complex< \_Tp > **sin** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **sinh** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **sqrt** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **tan** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **tanh** (const complex< \_Tp > &)
- constexpr float **tgamma** (float \_\_x)
- constexpr float **trunc** (float \_\_x)

### 6.587.1 Detailed Description

This is a Standard C++ Library header.

### 6.587.2 Function Documentation

#### abs()

```
template<typename _Tp>
_Tp std::abs (
    const complex< _Tp > & __z) [inline]
```

Return magnitude of z.

**cos()**

```
template<typename _Tp>
complex< _Tp > std::cos (
    const complex< _Tp > & __z) [inline]
```

Return complex cosine of  $z$ .

**cosh()**

```
template<typename _Tp>
complex< _Tp > std::cosh (
    const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic cosine of  $z$ .

**exp()**

```
template<typename _Tp>
complex< _Tp > std::exp (
    const complex< _Tp > & __z) [inline]
```

Return complex base  $e$  exponential of  $z$ .

**fabs()**

```
template<typename _Tp>
_Tp std::fabs (
    const std::complex< _Tp > & __z) [inline]
fabs(__z) TR1 8.1.8 [tr.c99.cmplx.fabs]
```

**log()**

```
template<typename _Tp>
complex< _Tp > std::log (
    const complex< _Tp > & __z) [inline]
```

Return complex natural logarithm of  $z$ .

**log10()**

```
template<typename _Tp>
complex< _Tp > std::log10 (
    const complex< _Tp > & __z) [inline]
```

Return complex base 10 logarithm of  $z$ .

**pow()**

```
template<typename _Tp>
complex< _Tp > std::pow (
    const complex< _Tp > & __z,
    int __n) [inline]
```

Return  $x$  to the  $y$ 'th power.

**sin()**

```
template<typename _Tp>
complex< _Tp > std::sin (
    const complex< _Tp > & __z) [inline]
```

Return complex sine of  $z$ .



**sinh()**

```
template<typename _Tp>
complex< _Tp > std::sinh (
    const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic sine of z.

**sqrt()**

```
template<typename _Tp>
complex< _Tp > std::sqrt (
    const complex< _Tp > & __z) [inline]
```

Return complex square root of z.

**tan()**

```
template<typename _Tp>
complex< _Tp > std::tan (
    const complex< _Tp > & __z) [inline]
```

Return complex tangent of z.

**tanh()**

```
template<typename _Tp>
complex< _Tp > std::tanh (
    const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic tangent of z.

## 6.588 mdspan File Reference

**Macros**

- #define `__glibcxx_want_aligned_accessor`
- #define `__glibcxx_want_mdspan`
- #define `_GLIBCXX_MDSPAN`

### 6.588.1 Detailed Description

This is a Standard C++ Library header.

## 6.589 experimental/memory File Reference

**Namespaces**

- namespace `std`
- namespace `std::experimental`

**Macros**

- #define `__cpp_lib_experimental_observer_ptr`
- #define `_GLIBCXX_EXPERIMENTAL_MEMORY`

## Functions

- `template<typename _Tp>`  
`constexpr observer_ptr< _Tp > std::experimental::make_observer (_Tp *__p) noexcept`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator!= (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator!= (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp>`  
`constexpr void std::experimental::swap (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept`

### 6.589.1 Detailed Description

This is a TS C++ Library header.

## 6.590 ext/memory File Reference

### Classes

- class [\\_\\_gnu\\_cxx::Temporary\\_buffer<\\_ForwardIterator, \\_Tp>](#)
- struct [\\_\\_gnu\\_cxx::temporary\\_buffer<\\_ForwardIterator, \\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Macros

- `#define _EXT_MEMORY`

## Functions

- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n (_InputIter __first, _Size __count, ↵  
_ForwardIter __result)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n (_InputIter __first, _Size __count, ↵  
_ForwardIter __result, std::input\_iterator\_tag)`

- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter>`  
`std::pair< _RandomAccessIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n ( _RandomAccessIter __↵`  
`__first, _Size __count, _ForwardIter __result, std::random_access_iterator_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator>`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a ( _InputIter __first, _Size __count,`  
`_ForwardIter __result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp>`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a ( _InputIter __first, _Size __count,`  
`_ForwardIter __result, std::allocator< _Tp >)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n ( _InputIter __first, _Size __count, __↵`  
`ForwardIter __result)`

### 6.590.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.591 memory File Reference

### Namespaces

- namespace `std`

### Macros

- `#define __glibcxx_want_addressof_constexpr`
- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_assume_aligned`
- `#define __glibcxx_want_atomic_shared_ptr`
- `#define __glibcxx_want_atomic_value_initialization`
- `#define __glibcxx_want_constexpr_dynamic_alloc`
- `#define __glibcxx_want_constexpr_memory`
- `#define __glibcxx_want_enable_shared_from_this`
- `#define __glibcxx_want_indirect`
- `#define __glibcxx_want_is_sufficiently_aligned`
- `#define __glibcxx_want_make_unique`
- `#define __glibcxx_want_out_ptr`
- `#define __glibcxx_want_parallel_algorithm`
- `#define __glibcxx_want_polymorphic`
- `#define __glibcxx_want_ranges`
- `#define __glibcxx_want_raw_memory_algorithms`
- `#define __glibcxx_want_shared_ptr_arrays`
- `#define __glibcxx_want_shared_ptr_weak_type`
- `#define __glibcxx_want_smart_ptr_for_overwrite`
- `#define __glibcxx_want_smart_ptr_owner_equality`
- `#define __glibcxx_want_start_lifetime_as`
- `#define __glibcxx_want_to_address`
- `#define __glibcxx_want_transparent_operators`
- `#define _GLIBCXX_MEMORY`

### Enumerations

- enum class `std::pointer_safety` { `relaxed` , `preferred` , `strict` }

## Functions

- void [std::declare\\_no\\_pointers](#) (char \*, size\_t)
- void [std::declare\\_reachable](#) (void \*)
- [pointer\\_safety std::get\\_pointer\\_safety](#) () noexcept
- void [std::undeclare\\_no\\_pointers](#) (char \*, size\_t)
- template<typename \_Tp>  
\_Tp \* [std::undeclare\\_reachable](#) (\_Tp \* \_\_p)

### 6.591.1 Detailed Description

This is a Standard C++ Library header.

## 6.592 experimental/memory\_resource File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_memory_resources`
- `#define _GLIBCXX_EXPERIMENTAL_MEMORY_RESOURCE`
- `#define _GLIBCXX_MAX_ALIGN_MATCHES_MALLOC`

### Typedefs

- template<typename \_Alloc>  
using [std::experimental::fundamentals\\_v2::pmr::resource\\_adaptor](#)

### Functions

- memory\_resource \* [std::experimental::fundamentals\\_v2::pmr::get\\_default\\_resource](#) () noexcept
- memory\_resource \* [std::experimental::fundamentals\\_v2::pmr::new\\_delete\\_resource](#) () noexcept
- memory\_resource \* [std::experimental::fundamentals\\_v2::pmr::null\\_memory\\_resource](#) () noexcept
- bool [std::experimental::fundamentals\\_v2::pmr::operator!=](#) (const memory\_resource &\_\_a, const memory\_resource &\_\_b) noexcept
- template<class \_Tp1, class \_Tp2>  
bool [std::experimental::fundamentals\\_v2::pmr::operator!=](#) (const polymorphic\_allocator< \_Tp1 > &\_\_a, const polymorphic\_allocator< \_Tp2 > &\_\_b) noexcept
- bool [std::experimental::fundamentals\\_v2::pmr::operator==](#) (const memory\_resource &\_\_a, const memory\_resource &\_\_b) noexcept
- template<class \_Tp1, class \_Tp2>  
bool [std::experimental::fundamentals\\_v2::pmr::operator==](#) (const polymorphic\_allocator< \_Tp1 > &\_\_a, const polymorphic\_allocator< \_Tp2 > &\_\_b) noexcept
- memory\_resource \* [std::experimental::fundamentals\\_v2::pmr::set\\_default\\_resource](#) (memory\_resource \* \_\_r) noexcept

### 6.592.1 Detailed Description

This is a TS C++ Library header.

### 6.592.2 Function Documentation

#### **get\_default\_resource()**

```
memory_resource * std::experimental::fundamentals_v2::pmr::get_default_resource () [inline],  
[noexcept]
```

Get the current default resource.

#### **set\_default\_resource()**

```
memory_resource * std::experimental::fundamentals_v2::pmr::set_default_resource (  
    memory_resource * __r) [inline], [noexcept]
```

Change the default resource and return the previous one.

## 6.593 memory\_resource File Reference

### Classes

- class [std::pmr::monotonic\\_buffer\\_resource](#)
- struct [std::pmr::pool\\_options](#)
- class [std::pmr::unsynchronized\\_pool\\_resource](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_memory_resource`
- `#define __glibcxx_want_polymorphic_allocator`
- `#define _GLIBCXX_MEMORY_RESOURCE`

### Functions

- [memory\\_resource \\* std::pmr::get\\_default\\_resource \(\)](#) [noexcept]
- [memory\\_resource \\* std::pmr::new\\_delete\\_resource \(\)](#) [noexcept]
- [memory\\_resource \\* std::pmr::null\\_memory\\_resource \(\)](#) [noexcept]
- [memory\\_resource \\* std::pmr::set\\_default\\_resource \(memory\\_resource \\* \\_\\_r\)](#) [noexcept]

### 6.593.1 Detailed Description

This is a Standard C++ Library header.

This header declares the [pmr](#) (std::pmr) memory resources.

### 6.593.2 Function Documentation

#### **get\_default\_resource()**

```
memory_resource * std::pmr::get_default_resource () [noexcept]
```

Get the current default memory resource pointer.

#### **null\_memory\_resource()**

```
memory_resource * std::pmr::null_memory_resource () [nodiscard], [noexcept]
```

A [pmr::memory\\_resource](#) that always throws `bad_alloc`

**set\_default\_resource()**

```
memory_resource * std::pmr::set_default_resource (
    memory_resource * __r) [noexcept]
```

Replace the default memory resource pointer.

**6.594 mutex File Reference****Classes**

- struct [std::once\\_flag](#)
- class [std::recursive\\_mutex](#)
- class [std::recursive\\_timed\\_mutex](#)
- class [std::timed\\_mutex](#)

**Namespaces**

- namespace [std](#)

**Macros**

- `#define` [\\_\\_glibcxx\\_want\\_scoped\\_lock](#)
- `#define` [\\_GLIBCXX\\_MUTEX](#)

**Functions**

- void [std::\\_\\_once\\_proxy](#) (void)
- template<typename \_Callable, typename... \_Args>  
void [std::call\\_once](#) ([once\\_flag](#) &\_\_once, \_Callable &&\_\_f, \_Args &&... \_\_args)
- template<typename \_L1, typename \_L2, typename... \_L3>  
void [std::lock](#) (\_L1 &\_\_l1, \_L2 &\_\_l2, \_L3 &... \_\_l3)
- template<typename \_L1, typename \_L2, typename... \_L3>  
int [std::try\\_lock](#) (\_L1 &\_\_l1, \_L2 &\_\_l2, \_L3 &... \_\_l3)

**6.594.1 Detailed Description**

This is a Standard C++ Library header.

**6.595 numbers File Reference****Macros**

- `#define` [\\_\\_glibcxx\\_want\\_math\\_constants](#)
- `#define` [\\_GLIBCXX\\_NUMBERS](#)

**6.595.1 Detailed Description**

This is a Standard C++ Library header.

**6.596 experimental/numeric File Reference****Namespaces**

- namespace [std](#)
- namespace [std::experimental](#)

## Macros

- `#define __cpp_lib_experimental_gcd_lcm`
- `#define _GLIBCXX_EXPERIMENTAL_NUMERIC`

## Functions

- `template<typename _Mn, typename _Nn>`  
`constexpr common\_type\_t< _Mn, _Nn > std::experimental::gcd (_Mn __m, _Nn __n) noexcept`
- `template<typename _Mn, typename _Nn>`  
`constexpr common\_type\_t< _Mn, _Nn > std::experimental::lcm (_Mn __m, _Nn __n)`

### 6.596.1 Detailed Description

This is a TS C++ Library header.

## 6.597 [ext/numeric](#) File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _EXT_NUMERIC`

## Functions

- `template<typename _Tp, typename _Integer>`  
`_Tp \_\_gnu\_cxx::\_\_power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`  
`_Tp \_\_gnu\_cxx::\_\_power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void \_\_gnu\_cxx::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _Tp, typename _Integer>`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`

### 6.597.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.598 [numeric](#) File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

## Macros

- `#define __glibcxx_want_constexpr_numeric`
- `#define __glibcxx_want_gcd`
- `#define __glibcxx_want_gcd_lcm`

- `#define __glibcxx_want_interpolate`
- `#define __glibcxx_want_lcm`
- `#define __glibcxx_want_parallel_algorithm`
- `#define __glibcxx_want_ranges_iota`
- `#define __glibcxx_want_saturation_arithmetic`
- `#define _GLIBCXX_NUMERIC`
- `#define _PSTL_NUMERIC_FORWARD_DECLARED`

## Functions

- `template<typename _Res, typename _Tp>`  
`constexpr _Res std::detail::__abs_r (_Tp __val)`
- `template<typename>`  
`void std::detail::__abs_r (bool)=delete`
- `template<typename _Tp>`  
`constexpr _Tp std::detail::__gcd (_Tp __m, _Tp __n)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator std::exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵  
result, _Tp __init)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _OutputIterator std::exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵  
result, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵  
result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵  
result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>`  
`constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵  
result, _BinaryOperation __binary_op, _Tp __init)`
- `template<typename _InputIterator>`  
`constexpr iterator_traits< _InputIterator >::value_type std::reduce (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator std::transform_exclusive_scan (_InputIterator __first, _InputIterator __last, _Output↵  
Iterator __result, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator std::transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _Output↵  
Iterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>`  
`constexpr _OutputIterator std::transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _Output↵  
Iterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _Tp std::transform_reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation  
__binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp std::transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,  
_Tp __init)`



- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`  
`constexpr _Tp std::transform\_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`

### 6.598.1 Detailed Description

This is a Standard C++ Library header.

## 6.599 [parallel/numeric](#) File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Macros

- `#define \_GLIBCXX\_PARALLEL\_NUMERIC\_H`

### Functions

- `template<typename _RAIter, typename _Tp, typename _BinaryOperation>`  
`_Tp std::\_\_parallel::\_\_accumulate\_switch (_RAIter __begin, _RAIter __end, _Tp __init, _BinaryOperation __binary_op, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag>`  
`_Tp std::\_\_parallel::\_\_accumulate\_switch (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`_Tp std::\_\_parallel::\_\_accumulate\_switch (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator std::\_\_parallel::\_\_adjacent\_difference\_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::\_\_parallel::\_\_adjacent\_difference\_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2>`  
`_Tp std::\_\_parallel::\_\_inner\_product\_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::\_\_parallel::\_\_inner\_product\_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag, gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator std::\_\_parallel::\_\_partial\_sum\_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::\_\_parallel::\_\_partial\_sum\_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp std::\_\_parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp std::\_\_parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op,`  
[\\_\\_gnu\\_parallel::sequential\\_tag](#))
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,`  
[\\_\\_gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag)
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,`  
[\\_\\_gnu\\_parallel::sequential\\_tag](#))
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,  $\leftrightarrow$`   
`_BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,  $\leftrightarrow$`   
`_BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,  $\leftrightarrow$`   
`_BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2,`  
[\\_\\_gnu\\_parallel::Parallelism](#))
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2,`  
[\\_\\_gnu\\_parallel::sequential\\_tag](#))
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result,`  
[\\_\\_gnu\\_parallel::sequential\\_tag](#))
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result,  $\leftrightarrow$`   
`_BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result,  $\leftrightarrow$`   
`_BinaryOperation __binary_op)`

### 6.599.1 Detailed Description

Parallel STL function calls corresponding to `stl_numeric.h`. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 6.600 experimental/optional File Reference

### Classes

- class [std::experimental::fundamentals\\_v1::bad\\_optional\\_access](#)
- struct [std::hash< experimental::optional< \\_Tp > >](#)
- struct [std::experimental::fundamentals\\_v1::in\\_place\\_t](#)
- struct [std::experimental::fundamentals\\_v1::nullopt\\_t](#)
- class [std::experimental::fundamentals\\_v1::optional< \\_Tp >](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_optional`
- `#define _GLIBCXX_EXPERIMENTAL_OPTIONAL`

### Variables

- constexpr [in\\_place\\_t std::experimental::in\\_place](#)
- constexpr [nullopt\\_t std::experimental::nullopt](#)

### 6.600.1 Detailed Description

This is a TS C++ Library header.

## 6.601 optional File Reference

### Macros

- `#define __glibcxx_want_constrained_equality`
- `#define __glibcxx_want_freestanding_optional`
- `#define __glibcxx_want_optional`
- `#define __glibcxx_want_optional_range_support`
- `#define _GLIBCXX_OPTIONAL`

### 6.601.1 Detailed Description

This is a Standard C++ Library header.

## 6.602 ostream File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define __glibcxx_want_print`
- `#define _GLIBCXX_OSTREAM`

## Functions

- `template<typename _CharT, typename _Traits>`  
[basic\\_ostream](#)< \_CharT, \_Traits > & [std::endl](#) ([basic\\_ostream](#)< \_CharT, \_Traits > & \_\_os)
- `template<typename _CharT, typename _Traits>`  
[basic\\_ostream](#)< \_CharT, \_Traits > & [std::ends](#) ([basic\\_ostream](#)< \_CharT, \_Traits > & \_\_os)
- `template<typename _CharT, typename _Traits>`  
[basic\\_ostream](#)< \_CharT, \_Traits > & [std::flush](#) ([basic\\_ostream](#)< \_CharT, \_Traits > & \_\_os)

### 6.602.1 Detailed Description

This is a Standard C++ Library header.

## 6.603 algo.h File Reference

### Classes

- struct [std::\\_\\_parallel::\\_\\_CRandNumber](#)< \_MustBeInt >

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Functions

- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>`  
[\\_FIterator](#) [std::\\_\\_parallel::\\_\\_adjacent\\_find\\_switch](#) ([\\_FIterator](#) \_\_begin, [\\_FIterator](#) \_\_end, [\\_BinaryPredicate](#) \_\_pred, [\\_IteratorTag](#))
- `template<typename _FIterator, typename _IteratorTag>`  
[\\_FIterator](#) [std::\\_\\_parallel::\\_\\_adjacent\\_find\\_switch](#) ([\\_FIterator](#) \_\_begin, [\\_FIterator](#) \_\_end, [\\_IteratorTag](#))
- `template<typename _RAIter, typename _BinaryPredicate>`  
[\\_RAIter](#) [std::\\_\\_parallel::\\_\\_adjacent\\_find\\_switch](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [\\_BinaryPredicate](#) \_\_pred, [random\\_access\\_iterator\\_tag](#))
- `template<typename _RAIter>`  
[\\_RAIter](#) [std::\\_\\_parallel::\\_\\_adjacent\\_find\\_switch](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [random\\_access\\_iterator\\_tag](#))
- `template<typename _Iter, typename _Predicate, typename _IteratorTag>`  
[iterator\\_traits](#)< [\\_Iter](#) >::difference\_type [std::\\_\\_parallel::\\_\\_count\\_if\\_switch](#) ([\\_Iter](#) \_\_begin, [\\_Iter](#) \_\_end, [\\_Predicate](#) \_\_pred, [\\_IteratorTag](#))
- `template<typename _RAIter, typename _Predicate>`  
[iterator\\_traits](#)< [\\_RAIter](#) >::difference\_type [std::\\_\\_parallel::\\_\\_count\\_if\\_switch](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [\\_Predicate](#) \_\_pred, [random\\_access\\_iterator\\_tag](#), [gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag)
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
[iterator\\_traits](#)< [\\_Iter](#) >::difference\_type [std::\\_\\_parallel::\\_\\_count\\_switch](#) ([\\_Iter](#) \_\_begin, [\\_Iter](#) \_\_end, const [\\_Tp](#) & \_\_value, [\\_IteratorTag](#))
- `template<typename _RAIter, typename _Tp>`  
[iterator\\_traits](#)< [\\_RAIter](#) >::difference\_type [std::\\_\\_parallel::\\_\\_count\\_switch](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, const [\\_Tp](#) & \_\_value, [random\\_access\\_iterator\\_tag](#), [gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag)

- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_Iter std::parallel::find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator`  
`__end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2>`  
`_Iter std::parallel::find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator`  
`__end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>`  
`_RAIter std::parallel::find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator`  
`__end2, _BinaryPredicate __comp, random\_access\_iterator\_tag, _IteratorTag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag>`  
`_Iter std::parallel::find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`_Iter std::parallel::find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp>`  
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag>`  
`_Function std::parallel::for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function>`  
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random\_access\_iterator\_tag,  
gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag>`  
`_OutputIterator std::parallel::generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator>`  
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random\_access\_iterator\_tag,  
gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag>`  
`void std::parallel::generate_switch (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator>`  
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random\_access\_iterator\_tag,  
gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag>`  
`_FIterator std::parallel::max_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator std::parallel::merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __result, _Compare __comp, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag>`  
`_FIterator std::parallel::min_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _FIterator, typename _Predicate, typename _IteratorTag>`  
`_FIterator std::parallel::partition_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag>`  
`void std::parallel::replace_if_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp>`  
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag>`  
`void std::parallel::replace_switch (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp>`  
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag>`  
`_FIterator std::parallel::search_n_switch (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_RAIter std::parallel::search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random\_access\_iterator\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2>`  
`_FIterator1 std::parallel::search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2>`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::set_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::set_intersection_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::set_symmetric_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAlter1, typename _RAlter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter std::parallel::set_union_switch (_RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _RAlter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAlter1, typename _RAlter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_RAlter2 std::parallel::transform1_switch (_RAlter1 __begin, _RAlter1 __end, _RAlter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAlter1, typename _RAlter2, typename _UnaryOperation>`  
`_RAlter2 std::parallel::transform1_switch (_RAlter1 __begin, _RAlter1 __end, _RAlter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3>`  
`_OutputIterator std::parallel::transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAlter1, typename _RAlter2, typename _RAlter3, typename _BinaryOperation>`  
`_RAlter3 std::parallel::transform2_switch (_RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _RAlter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator std::parallel::unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAlter, typename _RandomAccessOutputIterator, typename _Predicate>`  
`_RandomAccessOutputIterator std::parallel::unique_copy_switch (_RAlter __begin, _RAlter __last, _RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filterator>`  
`_Filterator std::parallel::adjacent_find (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator>`  
`_Filterator std::parallel::adjacent_find (_Filterator __begin, _Filterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _BinaryPredicate>`  
`_Filterator std::parallel::adjacent_find (_Filterator __begin, _Filterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _BinaryPredicate>`  
`_Filterator std::parallel::adjacent_find (_Filterator __begin, _Filterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::Parallelism __parallelism_tag)`



- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag )`
- `template<typename _Iter, typename _Tp>`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val )`
- `template<typename _Iter, typename _Tp>`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag )`
- `template<typename _Iter, typename _FIterator>`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2 )`
- `template<typename _Iter, typename _FIterator>`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag )`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate>`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag, _BinaryPredicate __comp )`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate>`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag )`
- `template<typename _Iter, typename _Predicate>`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred )`
- `template<typename _Iter, typename _Predicate>`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag )`
- `template<typename _Iter, typename _Function>`  
`_Function std::__parallel::for_each ( _Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag )`
- `template<typename _Iterator, typename _Function>`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f )`
- `template<typename _Iterator, typename _Function>`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism\_parallelism\_tag )`
- `template<typename _FIterator, typename _Generator>`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen )`
- `template<typename _FIterator, typename _Generator>`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism\_parallelism\_tag )`
- `template<typename _FIterator, typename _Generator>`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag )`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator std::__parallel::generate_n ( _OutputIterator __begin, _Size __n, _Generator __gen )`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator std::__parallel::generate_n ( _OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism\_parallelism\_tag )`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator std::__parallel::generate_n ( _OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag )`
- `template<typename _FIterator>`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end )`
- `template<typename _FIterator>`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism\_parallelism\_tag )`
- `template<typename _FIterator>`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag )`
- `template<typename _FIterator, typename _Compare>`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end, _Compare __comp )`



- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, _Compare __comp,`  
`__gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, __gnu_parallel::__Parallelism ↵`  
`__parallelism_tag)`
- `template<typename _Filterator>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, _Compare __comp)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, _Compare __comp,`  
`__gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _RAlter>`  
`void std::__parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end)`
- `template<typename _RAlter>`  
`void std::__parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter, typename _Compare>`  
`void std::__parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp)`
- `template<typename _RAlter, typename _Compare>`  
`void std::__parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _RAlter>`  
`void std::__parallel::partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end)`
- `template<typename _RAlter>`  
`void std::__parallel::partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter, typename _Compare>`  
`void std::__parallel::partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, _Compare __comp)`
- `template<typename _RAlter, typename _Compare>`  
`void std::__parallel::partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Predicate>`  
`_Filterator std::__parallel::partition (_Filterator __begin, _Filterator __end, _Predicate __pred)`

- `template<typename _FIterator, typename _Predicate>`  
`_FIterator std::parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void std::parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void std::parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void std::parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Tp>`  
`void std::parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Tp>`  
`void std::parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp>`  
`void std::parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp>`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp>`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp>`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2>`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2>`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`_ForwardIterator std::parallel::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`
- `template<typename _FIterator, typename _Integer, typename _Tp>`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp>`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator std::parallel::set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`



- `template<typename _RAIter>`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __↔  
__parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag  
__parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __↔  
parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism>`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __↔  
__parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __↔  
parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __↔  
__parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↔  
parallelism)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __end, _OutputIterator __result, _Unary↔  
Operation __unary_op)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __end, _OutputIterator __result, _Unary↔  
Operation __unary_op, __gnu_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator std::parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`

### 6.603.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algo.h` header.

The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 6.604 `algo.h` File Reference

### Namespaces

- namespace `std`
- namespace `std::parallel`

### Functions

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`bool std::parallel::equal_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::equal\_switch __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`bool std::parallel::equal_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`bool std::parallel::lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`bool std::parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`pair<_Iter1, _Iter2> std::parallel::mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2`  
`__begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2`  
`__begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_Filterator1 std::parallel::search_switch ( _Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2,`  
`_Filterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate>`  
`_RAIter1 std::parallel::search_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool std::parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool std::parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr bool std::parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _`  
`BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`bool std::parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate`  
`__binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool std::parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool std::parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::parallel::lexicographical_compare ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool std::parallel::lexicographical_compare ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2`  
`__end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool std::parallel::lexicographical_compare ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool std::parallel::lexicographical_compare ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2`  
`__end2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2`  
`__end2)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _↵`  
`Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _↵`  
`Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch ( _InputIterator1 __begin1, _InputIterator1 ↵`  
`__end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch ( _InputIterator1 __first1, _InputIterator1 ↵`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, __gnu_parallel::sequential_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch ( _InputIterator1 __first1, _InputIterator1 ↵`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate>`  
`constexpr _Filterator1 std::__parallel::search ( _Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2,`  
`_Filterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate>`  
`_Filterator1 std::__parallel::search ( _Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2`  
`__end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)`

### 6.604.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algobase.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 6.605 balanced\_quicksort.h File Reference

### Classes

- struct `__gnu_parallel::QSBThreadLocal< _RAIter >`

### Namespaces

- namespace `__gnu_parallel`

### Functions

- `template<typename _RAIter, typename _Compare>`  
`void __gnu_parallel::__parallel_sort_qsb ( _RAIter __begin, _RAIter __end, _Compare __comp, __ThreadIndex`  
`__num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`void __gnu_parallel::__qsb_conquer ( QSBThreadLocal< _RAIter > *__tls, _RAIter __begin, _RAIter __end,`  
`_Compare __comp, __ThreadIndex __iam, __ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare>`  
`std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__qsb_divide ( _RAIter __begin, _RAIter __end,`  
`_Compare __comp, __ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`void __gnu_parallel::__qsb_local_sort_with_helping ( QSBThreadLocal< _RAIter > *__tls, _Compare &__↵`  
`comp, __ThreadIndex __iam, bool __wait)`



### 6.605.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort.

It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

## 6.606 base.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_\\_binder1st<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_binder2nd<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_unary\\_negate<\\_Predicate, argument\\_type>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_EqualFromLess<\\_T1, \\_T2, \\_Compare>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_EqualTo<\\_T1, \\_T2>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Less<\\_T1, \\_T2>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Multiplies<\\_Tp1, \\_Tp2, \\_Result>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Plus<\\_Tp1, \\_Tp2, \\_Result>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequence<\\_Tp, \\_DifferenceTp>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequenceIterator<\\_Tp, \\_DifferenceTp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)
- namespace [\\_\\_gnu\\_sequential](#)
- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Macros

- `#define \_GLIBCXX\_PARALLEL\_ASSERT(_Condition)`

### Functions

- void [\\_\\_gnu\\_parallel::\\_\\_decode2](#) ([\\_CASable](#) \_\_x, int &\_\_a, int &\_\_b)
- [\\_CASable](#) [\\_\\_gnu\\_parallel::\\_\\_encode2](#) (int \_\_a, int \_\_b)
- [\\_ThreadIndex](#) [\\_\\_gnu\\_parallel::\\_\\_get\\_max\\_threads](#) ()
- bool [\\_\\_gnu\\_parallel::\\_\\_is\\_parallel](#) (const [\\_Parallelism](#) \_\_p)
- template<typename [\\_RAIter](#), typename [\\_Compare](#)>  
[\\_RAIter](#) [\\_\\_gnu\\_parallel::\\_\\_median\\_of\\_three\\_iterators](#) ([\\_RAIter](#) \_\_a, [\\_RAIter](#) \_\_b, [\\_RAIter](#) \_\_c, [\\_Compare](#) \_\_comp)
- template<typename [\\_Size](#)>  
[\\_Size](#) [\\_\\_gnu\\_parallel::\\_\\_rd\\_log2](#) ([\\_Size](#) \_\_n)
- template<typename [\\_Tp](#)>  
const [\\_Tp](#) & [\\_\\_gnu\\_parallel::max](#) (const [\\_Tp](#) &\_\_a, const [\\_Tp](#) &\_\_b)
- template<typename [\\_Tp](#)>  
const [\\_Tp](#) & [\\_\\_gnu\\_parallel::min](#) (const [\\_Tp](#) &\_\_a, const [\\_Tp](#) &\_\_b)

### 6.606.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.



## 6.607 `basic_iterator.h` File Reference

### 6.607.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 6.608 `checkers.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _Compare>`  
`bool \_\_gnu\_parallel::\_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __comp)`

### 6.608.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

## 6.609 `compiletime_settings.h` File Reference

### Macros

- `#define \_GLIBCXX\_CALL(__n)`
- `#define \_GLIBCXX\_PARALLEL\_ASSERTIONS`
- `#define \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_L1`
- `#define \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_TLB`
- `#define \_GLIBCXX\_SCALE\_DOWN\_FPU`
- `#define \_GLIBCXX\_VERBOSE\_LEVEL`

### 6.609.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

### 6.609.2 Macro Definition Documentation

#### `_GLIBCXX_CALL`

```
#define _GLIBCXX_CALL(  
    __n)
```

Macro to produce log message when entering a function.

#### Parameters

<code><a href="#">_↔</a></code>	Input size.
<code><a href="#">_n</a></code>	

See also

`_GLIBCXX_VERBOSE_LEVEL`

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__find_template()`, `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__merge_advance()`, `__gnu_parallel::__parallel_n`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__parallel_unique_copy()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::__multiseq_partition()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::multiway_merge()`, `__gnu_parallel::multiway_merge_3_variant()`, `__gnu_parallel::multiway_merge_4_variant()`, `__gnu_parallel::multiway_merge_loser_tree()`, `__gnu_parallel::multiway_merge_loser_tree_sentinel()`, `__gnu_parallel::multiway_merge_lo`, `__gnu_parallel::multiway_merge_sentinels()`, `__gnu_parallel::parallel_multiway_merge()`, and `__gnu_parallel::parallel_sort_mwms()`.

### `_GLIBCXX_PARALLEL_ASSERTIONS`

```
#define _GLIBCXX_PARALLEL_ASSERTIONS
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

### `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Consider the size of the L1 cache for `gnu_↵`  
`parallel::__parallel_random_shuffle()`.

### `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Consider the size of the TLB for `gnu_parallel_↵`  
`::__parallel_random_shuffle()`.

### `_GLIBCXX_SCALE_DOWN_FPU`

```
#define _GLIBCXX_SCALE_DOWN_FPU
```

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

### `_GLIBCXX_VERBOSE_LEVEL`

```
#define _GLIBCXX_VERBOSE_LEVEL
```

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

## 6.610 `equally_split.h` File Reference

### Namespaces

- namespace `__gnu_parallel`

### Functions

- `template<typename _DifferenceType, typename _OutputIterator>`  
`_OutputIterator` `__gnu_parallel::__equally_split` (`_DifferenceType` `__n`, `_ThreadIndex` `__num_threads`, `_Output_↵`  
`Iterator` `__s`)

- `template<typename _DifferenceType>  
_DifferenceType __gnu_parallel::__equally_split_point (_DifferenceType __n, __ThreadIndex __num_threads,  
__ThreadIndex __thread_no)`

### 6.610.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 6.611 features.h File Reference

### Macros

- `#define _GLIBCXX_BAL_QUICKSORT`
- `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
- `#define _GLIBCXX_FIND_EQUAL_SPLIT`
- `#define _GLIBCXX_FIND_GROWING_BLOCKS`
- `#define _GLIBCXX_MERGESORT`
- `#define _GLIBCXX_QUICKSORT`
- `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`
- `#define _GLIBCXX_TREE_FULL_COPY`
- `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

### 6.611.1 Detailed Description

Defines on whether to include algorithm variants.

Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

### 6.611.2 Macro Definition Documentation

#### `_GLIBCXX_BAL_QUICKSORT`

```
#define _GLIBCXX_BAL_QUICKSORT
```

Include parallel dynamically load-balanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

#### `_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

```
#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
```

Include the equal-sized blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

#### `_GLIBCXX_FIND_EQUAL_SPLIT`

```
#define _GLIBCXX_FIND_EQUAL_SPLIT
```

Include the equal splitting variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

**\_GLIBCXX\_FIND\_GROWING\_BLOCKS**

```
#define _GLIBCXX_FIND_GROWING_BLOCKS
```

Include the growing blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

**\_GLIBCXX\_MERGESORT**

```
#define _GLIBCXX_MERGESORT
```

Include parallel multi-way mergesort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

**\_GLIBCXX\_QUICKSORT**

```
#define _GLIBCXX_QUICKSORT
```

Include parallel unbalanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

**\_GLIBCXX\_TREE\_DYNAMIC\_BALANCING**

```
#define _GLIBCXX_TREE_DYNAMIC_BALANCING
```

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

**\_GLIBCXX\_TREE\_FULL\_COPY**

```
#define _GLIBCXX_TREE_FULL_COPY
```

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

See also

`__gnu_parallel::_Rb_tree`

**\_GLIBCXX\_TREE\_INITIAL\_SPLITTING**

```
#define _GLIBCXX_TREE_INITIAL_SPLITTING
```

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

## 6.612 find.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
[std::pair](#)< \_RAIter1, \_RAIter2 > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, [↔](#)  
 \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
[std::pair](#)< \_RAIter1, \_RAIter2 > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, [↔](#)  
 \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, [constant\\_size\\_blocks\\_tag](#))
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
[std::pair](#)< \_RAIter1, \_RAIter2 > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, [↔](#)  
 \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, [equal\\_split\\_tag](#))
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
[std::pair](#)< \_RAIter1, \_RAIter2 > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, [↔](#)  
 \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, [growing\\_blocks\\_tag](#))

### 6.612.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

## 6.613 find\_selectors.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector](#)< \_Filterator >
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_find\\_selector](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_mismatch\\_selector](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.613.1 Detailed Description

\_Function objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

## 6.614 for\_each.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>`  
[\\_UserOp](#) [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access](#) (\_Iter \_\_begin, \_Iter \_\_end, [\\_UserOp](#) \_\_user $\leftrightarrow$   
 \_\_op, \_Functionality & \_\_functionality, \_Red \_\_reduction, \_Result \_\_reduction\_start, \_Result & \_\_output, typename  
[std::iterator\\_traits](#)< \_Iter >::difference\_type \_\_bound, [\\_Parallelism](#) \_\_parallelism\_tag)

### 6.614.1 Detailed Description

Main interface for embarrassingly parallel functions.

The explicit implementation are in other header files, like `workstealing.h`, `par_loop.h`, `omp_loop.h`, and `omp_loop_↵static.h`. This file is a GNU parallel extension to the Standard C++ Library.

## 6.615 for\_each\_selectors.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_binop\\_reduct< \\_BinOp >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_selector< \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_difference\\_selector< \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_if\\_selector< \\_It, \\_Diff >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_selector< \\_It, \\_Diff >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_fill\\_selector< \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_selector< \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_generate\\_selector< \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_for\\_each\\_selector< \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_identity\\_selector< \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_inner\\_product\\_selector< \\_It, \\_It2, \\_Tp >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_max\\_element\\_reduct< \\_Compare, \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_min\\_element\\_reduct< \\_Compare, \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_replace\\_if\\_selector< \\_It, \\_Op, \\_Tp >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_replace\\_selector< \\_It, \\_Tp >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_transform1\\_selector< \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_transform2\\_selector< \\_It >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_DummyReduct](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Nothing](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.615.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

## 6.616 iterator.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_\\_IteratorPair< \\_Iterator1, \\_Iterator2, \\_IteratorCategory >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_IteratorTriple< \\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory >](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.616.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

## 6.617 list\_partition.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter>  
void \_\_gnu\_parallel::\_\_shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter>  
void \_\_gnu\_parallel::\_\_shrink\_and\_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FuncType>  
size_t \_\_gnu\_parallel::list\_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FuncType & __f, int __oversampling=0)`

#### 6.617.1 Detailed Description

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 6.618 losertree.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >::\\_Loser](#)
- struct [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >::\\_Loser](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTree< \\_\\_stable, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTree< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointer< \\_\\_stable, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguardedBase< \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguardedBase< \\_Tp, \\_Compare >](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

#### 6.618.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

## 6.619 merge.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- template<typename \_RAIter1, typename \_RAIter2, typename \_OutputIterator, typename \_DifferenceTp, typename \_Compare>  
\_OutputIterator [\\_\\_gnu\\_parallel::merge\\_advance](#) (\_RAIter1 &\_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 &\_\_begin2, \_RAIter2 \_\_end2, \_OutputIterator \_\_target, \_DifferenceTp \_\_max\_length, \_Compare \_\_comp)
- template<typename \_RAIter1, typename \_RAIter2, typename \_OutputIterator, typename \_DifferenceTp, typename \_Compare>  
\_OutputIterator [\\_\\_gnu\\_parallel::merge\\_advance\\_movc](#) (\_RAIter1 &\_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 &\_\_begin2, \_RAIter2 \_\_end2, \_OutputIterator \_\_target, \_DifferenceTp \_\_max\_length, \_Compare \_\_comp)
- template<typename \_RAIter1, typename \_RAIter2, typename \_OutputIterator, typename \_DifferenceTp, typename \_Compare>  
\_OutputIterator [\\_\\_gnu\\_parallel::merge\\_advance\\_usual](#) (\_RAIter1 &\_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 &\_\_begin2, \_RAIter2 \_\_end2, \_OutputIterator \_\_target, \_DifferenceTp \_\_max\_length, \_Compare \_\_comp)
- template<typename \_RAIter1, typename \_RAIter3, typename \_Compare>  
\_RAIter3 [\\_\\_gnu\\_parallel::parallel\\_merge\\_advance](#) (\_RAIter1 &\_\_begin1, \_RAIter1 \_\_end1, \_RAIter1 &\_\_begin2, \_RAIter1 \_\_end2, \_RAIter3 \_\_target, typename [std::iterator\\_traits](#)< \_RAIter1 >::difference\_type \_\_max\_length, \_Compare \_\_comp)
- template<typename \_RAIter1, typename \_RAIter2, typename \_RAIter3, typename \_Compare>  
\_RAIter3 [\\_\\_gnu\\_parallel::parallel\\_merge\\_advance](#) (\_RAIter1 &\_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 &\_\_begin2, \_RAIter2 \_\_end2, \_RAIter3 \_\_target, typename [std::iterator\\_traits](#)< \_RAIter1 >::difference\_type \_\_max\_length, \_Compare \_\_comp)

### 6.619.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

## 6.620 multiseq\_selection.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::Lexicographic](#)< \_T1, \_T2, \_Compare >
- class [\\_\\_gnu\\_parallel::LexicographicReverse](#)< \_T1, \_T2, \_Compare >

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define __S(__i)`
- `#define __S(__i)`

## Functions

- template<typename \_RanSeqs, typename \_RankType, typename \_RankIterator, typename \_Compare>  
void [\\_\\_gnu\\_parallel::multiseq\\_partition](#) (\_RanSeqs \_\_begin\_seqs, \_RanSeqs \_\_end\_seqs, \_RankType \_\_rank, \_RankIterator \_\_begin\_offsets, \_Compare \_\_comp=[std::less](#)< typename [std::iterator\\_traits](#)< typename [std::iterator\\_traits](#)< \_RanSeqs >::value\_type::first\_type >::value\_type >())
- template<typename \_Tp, typename \_RanSeqs, typename \_RankType, typename \_Compare>  
\_Tp [\\_\\_gnu\\_parallel::multiseq\\_selection](#) (\_RanSeqs \_\_begin\_seqs, \_RanSeqs \_\_end\_seqs, \_RankType \_\_rank, \_RankType &\_\_offset, \_Compare \_\_comp=[std::less](#)< \_Tp >())

### 6.620.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets.

The algorithm description can be found in



P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. *Journal of Parallel and Distributed Computing*, 12(2):171-177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

## 6.621 multiway\_merge.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch](#)< \_\_sentinels, \_RAlterIterator, \_RAlter3, \_DifferenceTp, \_Compare >
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch](#)< true, \_RAlterIterator, \_RAlter3, \_DifferenceTp, \_Compare >
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch](#)< \_\_sentinels, \_RAlterIterator, \_RAlter3, \_DifferenceTp, \_Compare >
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch](#)< true, \_RAlterIterator, \_RAlter3, \_DifferenceTp, \_Compare >
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)< \_\_sentinels, \_\_stable, \_RAlterIterator, \_RAlter3, \_DifferenceTp, \_Compare >
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)< false, \_\_stable, \_RAlterIterator, \_RAlter3, \_DifferenceTp, \_Compare >
- class [\\_\\_gnu\\_parallel::\\_GuardedIterator](#)< \_RAlter, \_Compare >
- struct [\\_\\_gnu\\_parallel::\\_LoserTreeTraits](#)< \_Tp >
- struct [\\_\\_gnu\\_parallel::\\_SamplingSorter](#)< \_\_stable, \_RAlter, \_StrictWeakOrdering >
- struct [\\_\\_gnu\\_parallel::\\_SamplingSorter](#)< false, \_RAlter, \_StrictWeakOrdering >

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define GLIBCXX\_PARALLEL\_DECISION(__a, __b, __c, __d)`
- `#define GLIBCXX\_PARALLEL\_LENGTH(__s)`
- `#define GLIBCXX\_PARALLEL\_MERGE\_3\_CASE(__a, __b, __c, __c0, __c1)`
- `#define GLIBCXX\_PARALLEL\_MERGE\_4\_CASE(__a, __b, __c, __d, __c0, __c1, __c2)`

### Functions

- `template<typename _RAlter1, typename _RAlter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>  
_OutputIterator \_\_gnu\_parallel::merge\_advance (_RAlter1 &__begin1, _RAlter1 __end1, _RAlter2 &__begin2, _RAlter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<bool __stable, bool __sentinels, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare>  
_RAlter3 \_\_gnu\_parallel::sequential\_multiway\_merge (_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAlterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare>  
_RAlterOut \_\_gnu\_parallel::multiway\_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare>  
_RAlterOut \_\_gnu\_parallel::multiway\_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare>  
_RAlterOut \_\_gnu\_parallel::multiway\_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare>  
_RAlterOut \_\_gnu\_parallel::multiway\_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::default\_parallel\_tag __tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare>  
_RAlterOut \_\_gnu\_parallel::multiway\_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::parallel\_tag __tag=\_\_gnu\_parallel::parallel\_tag(0))`

- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_3\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_4\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>  
void \_\_gnu\_parallel::multiway\_merge\_exact\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>  
void \_\_gnu\_parallel::multiway\_merge\_sampling\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,  
_RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,  
_RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,  
_RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,  
_RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,  
_RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::parallel\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, ThreadIndex __num_threads)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,  
_RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,  
_RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPair↵`  
`Iterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __↵`  
`tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter↵`  
`PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag`  
`__tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter↵`  
`PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter↵`  
`PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag`  
`__tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter↵`  
`PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __↵`  
`tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter↵`  
`PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`

### 6.621.1 Detailed Description

Implementation of sequential and parallel multiway merge.

Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

### 6.621.2 Macro Definition Documentation

#### `_GLIBCXX_PARALLEL_LENGTH`

```
#define _GLIBCXX_PARALLEL_LENGTH(  
    __s)
```

Length of a sequence described by a pair of iterators.

Referenced by [gnu\\_parallel::sequential\\_multiway\\_merge\(\)](#), [gnu\\_parallel::multiway\\_merge\\_exact\\_splitting\(\)](#),  
[gnu\\_parallel::multiway\\_merge\\_loser\\_tree\(\)](#), [gnu\\_parallel::multiway\\_merge\\_sampling\\_splitting\(\)](#), and [gnu\\_parallel::parallel\\_multiway\\_merge\(\)](#).

## 6.622 multiway\_mergesort.h File Reference

### Classes

- struct [gnu\\_parallel::Piece<\\_DifferenceTp>](#)
- struct [gnu\\_parallel::PMWMSSortingData<\\_RAIter>](#)
- struct [gnu\\_parallel::SplitConsistently<\\_\\_exact, \\_RAIter, \\_Compare, \\_SortingPlacesIterator>](#)
- struct [gnu\\_parallel::SplitConsistently<false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator>](#)
- struct [gnu\\_parallel::SplitConsistently<true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator>](#)

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _DifferenceTp>`  
`void \_\_gnu\_parallel::\_\_determine\_samples (\_PMWSSortingData< _RAIter > *__sd, _DifferenceTp __num↔  
 _samples)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::parallel\_sort\_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex  
 __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::parallel\_sort\_mwms\_pu (\_PMWSSortingData< _RAIter > *__sd, _Compare &__comp)`

### 6.622.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

## 6.623 numericfwd.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Functions

- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag>`  
`_Tp std::\_\_parallel::\_\_accumulate\_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _Iter, typename _Tp, typename _Tag>`  
`_Tp std::\_\_parallel::\_\_accumulate\_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper>`  
`_Tp std::\_\_parallel::\_\_accumulate\_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random\_access\_iterator\_tag,  
\_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>`  
`_OIter std::\_\_parallel::\_\_adjacent\_difference\_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter std::\_\_parallel::\_\_adjacent\_difference\_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1,  
 typename _Tag2>`  
`_Tp std::\_\_parallel::\_\_inner\_product\_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _Binary↔  
 Function2, _Tag1, _Tag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::\_\_parallel::\_\_inner\_product\_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _Binary↔  
 Function2, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism=\_\_gnu\_parallel::parallel\_unbal`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>`  
`_OIter std::\_\_parallel::\_\_partial\_sum\_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter std::\_\_parallel::\_\_partial\_sum\_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::\_\_parallel::accumulate (_Iter, _Iter, _Tp)`

- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Olter>`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter)`
- `template<typename _Iter, typename _Olter>`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Olter>`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper>`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, _BinaryOper)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper>`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper>`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Olter>`  
`_Olter std::__parallel::partial_sum (_Iter, _Iter, _Olter __result)`
- `template<typename _Iter, typename _Olter>`  
`_Olter std::__parallel::partial_sum (_Iter, _Iter, _Olter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper>`  
`_Olter std::__parallel::partial_sum (_Iter, _Iter, _Olter, _BinaryOper)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper>`  
`_Olter std::__parallel::partial_sum (_Iter, _Iter, _Olter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`

### 6.623.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 6.624 omp\_loop.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
_Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits<_RAIter>::difference_type __bound)`

### 6.624.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

## 6.625 omp\_loop\_static.h File Reference

### Namespaces

- namespace `__gnu_parallel`

## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits<_RAIter>::difference_type __bound)`

### 6.625.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

## 6.626 par\_loop.h File Reference

### Namespaces

- namespace `__gnu_parallel`

## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
_Op __gnu_parallel::__for_each_template_random_access_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits<_RAIter>::difference_type __bound)`

### 6.626.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

## 6.627 parallel.h File Reference

### 6.627.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

## 6.628 `partial_sum.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum (_Iter __begin, _Iter __end, _OutputIterator __result, ↵`  
`_BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_basecase (_Iter __begin, _Iter __end, _OutputIterator`  
`__result, _BinaryOperation __bin_op, typename std::iterator\_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _OutputIterator __↵`  
`result, _BinaryOperation __bin_op, typename std::iterator\_traits< _Iter >::difference_type __n)`

#### 6.628.1 Detailed Description

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

## 6.629 `partition.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define \_GLIBCXX\_VOLATILE`

### Functions

- `template<typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_nth\_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __↵`  
`comp)`
- `template<typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_partial\_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __↵`  
`comp)`
- `template<typename _RAIter, typename _Predicate>`  
`std::iterator\_traits< _RAIter >::difference_type \_\_gnu\_parallel::\_\_parallel\_partition (_RAIter __begin, _RAIter ↵`  
`__end, _Predicate __pred, \_ThreadIndex __num_threads)`

#### 6.629.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

#### 6.629.2 Macro Definition Documentation

##### `_GLIBCXX_VOLATILE`

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variables volatile.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partition\(\)](#).

## 6.630 queue.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define` [\\_GLIBCXX\\_VOLATILE](#)

#### 6.630.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

#### 6.630.2 Macro Definition Documentation

##### [\\_GLIBCXX\\_VOLATILE](#)

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variable volatile in this file.

## 6.631 quicksort.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Compare>`  
void [\\_\\_gnu\\_parallel::\\_parallel\\_sort\\_qs](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, [\\_ThreadIndex](#) \_\_num\_threads)
- `template<typename _RAIter, typename _Compare>`  
void [\\_\\_gnu\\_parallel::\\_parallel\\_sort\\_qs\\_conquer](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, [\\_ThreadIndex](#) \_\_num\_threads)
- `template<typename _RAIter, typename _Compare>`  
[std::iterator\\_traits<\\_RAIter>::difference\\_type](#) [\\_\\_gnu\\_parallel::\\_parallel\\_sort\\_qs\\_divide](#) (\_RAIter \_\_begin, [\\_ThreadIndex](#) \_\_num\_threads, \_RAIter \_\_end, \_Compare \_\_comp, [std::iterator\\_traits<\\_RAIter>::difference\\_type](#) \_\_pivot\_rank, [std::iterator\\_traits<\\_RAIter>::difference\\_type](#) \_\_num\_samples, [\\_ThreadIndex](#) \_\_num\_threads)

#### 6.631.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

## 6.632 random\_number.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_RandomNumber](#)



## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.632.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

## 6.633 random\_shuffle.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter >](#)
- struct [\\_\\_gnu\\_parallel::\\_DRSSorterPU<\\_RAIter, \\_RandomNumberGenerator >](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Typedefs

- typedef unsigned short [\\_\\_gnu\\_parallel::\\_BinIndex](#)

### Functions

- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator \_\_rng= [\\_RandomNumber\(\)](#))
- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle\\_drs](#) (\_RAIter \_\_begin, \_RAIter \_\_end, typename [std::iterator\\_traits<\\_RAIter >::difference\\_type](#) \_\_n, [\\_ThreadIndex](#) \_\_num\_threads, \_RandomNumberGenerator &\_\_rng)
- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle\\_drs\\_pu](#) ([\\_DRSSorterPU](#)<\_RAIter, \_RandomNumberGenerator > \*\_\_pus)
- template<typename \_RandomNumberGenerator>  
int [\\_\\_gnu\\_parallel::\\_random\\_number\\_pow2](#) (int \_\_logp, \_RandomNumberGenerator &\_\_rng)
- template<typename \_Tp>  
[\\_Tp](#) [\\_\\_gnu\\_parallel::\\_round\\_up\\_to\\_pow2](#) (\_Tp \_\_x)
- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void [\\_\\_gnu\\_parallel::\\_sequential\\_random\\_shuffle](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator &\_\_rng)

### 6.633.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

## 6.634 search.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- template<typename \_RAIter, typename \_DifferenceTp>  
void [\\_\\_gnu\\_parallel::\\_\\_calc\\_borders](#) (\_RAIter \_\_elements, \_DifferenceTp \_\_length, \_DifferenceTp \*\_\_off)
- template<typename \_\_RAIter1, typename \_\_RAIter2, typename \_Pred>  
\_\_RAIter1 [\\_\\_gnu\\_parallel::\\_\\_search\\_template](#) (\_\_RAIter1 \_\_begin1, \_\_RAIter1 \_\_end1, \_\_RAIter2 \_\_begin2,  $\leftrightarrow$  \_\_RAIter2 \_\_end2, \_Pred \_\_pred)

### 6.634.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

## 6.635 set\_operations.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_Iter, typename \_OutputIterator>  
\_OutputIterator [\\_\\_gnu\\_parallel::\\_\\_copy\\_tail](#) (std::pair< \_Iter, \_Iter > \_\_b, std::pair< \_Iter, \_Iter > \_\_e,  $\leftrightarrow$  \_OutputIterator \_\_r)
- template<typename \_Iter, typename \_OutputIterator, typename \_Compare>  
\_OutputIterator [\\_\\_gnu\\_parallel::\\_\\_parallel\\_set\\_difference](#) (\_Iter \_\_begin1, \_Iter \_\_end1, \_Iter \_\_begin2,  $\leftrightarrow$  \_Iter \_\_end2, \_OutputIterator \_\_result, \_Compare \_\_comp)
- template<typename \_Iter, typename \_OutputIterator, typename \_Compare>  
\_OutputIterator [\\_\\_gnu\\_parallel::\\_\\_parallel\\_set\\_intersection](#) (\_Iter \_\_begin1, \_Iter \_\_end1, \_Iter \_\_begin2,  $\leftrightarrow$  \_Iter \_\_end2, \_OutputIterator \_\_result, \_Compare \_\_comp)
- template<typename \_Iter, typename \_OutputIterator, typename \_Operation>  
\_OutputIterator [\\_\\_gnu\\_parallel::\\_\\_parallel\\_set\\_operation](#) (\_Iter \_\_begin1, \_Iter \_\_end1, \_Iter \_\_begin2,  $\leftrightarrow$  \_Iter \_\_end2, \_OutputIterator \_\_result, \_Operation \_\_op)
- template<typename \_Iter, typename \_OutputIterator, typename \_Compare>  
\_OutputIterator [\\_\\_gnu\\_parallel::\\_\\_parallel\\_set\\_symmetric\\_difference](#) (\_Iter \_\_begin1, \_Iter \_\_end1, \_Iter \_\_begin2, \_Iter \_\_end2, \_OutputIterator \_\_result, \_Compare \_\_comp)
- template<typename \_Iter, typename \_OutputIterator, typename \_Compare>  
\_OutputIterator [\\_\\_gnu\\_parallel::\\_\\_parallel\\_set\\_union](#) (\_Iter \_\_begin1, \_Iter \_\_end1, \_Iter \_\_begin2, \_Iter \_\_end2, \_OutputIterator \_\_result, \_Compare \_\_comp)

### 6.635.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 6.636 settings.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_Settings](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Macros

- `#define _GLIBCXX_PARALLEL_CONDITION(__c)`

### 6.636.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

### 6.636.2 Deciding whether to run an algorithm in parallel.

There are several ways the user can switch on and off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. It is a tristate variable corresponding to:

- a. `force_sequential`, meaning the sequential algorithm is executed.
- b. `force_parallel`, meaning the parallel algorithm is executed.
- c. `heuristic`

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

### 6.636.3 Macro Definition Documentation

#### `_GLIBCXX_PARALLEL_CONDITION`

```
#define _GLIBCXX_PARALLEL_CONDITION(  
    __c)
```

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

#### Parameters

<code>__c</code>	A condition that is convertible to bool that is overruled by <code>__gnu_parallel::_Settings::algorithm_strategy</code> .
<code>__c</code>	Usually a decision based on the input size.

## 6.637 sort.h File Reference

### Namespaces

- namespace `__gnu_parallel`

## Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename \_Parallelism>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_Parallelism __↵  
parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced\_quicksort\_tag  
__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default\_parallel\_tag  
__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_exact\_tag  
__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_sampling\_tag  
__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_tag  
__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel\_tag __↵  
parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort\_tag __↵  
parallelism)`

### 6.637.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

## 6.638 tags.h File Reference

### Classes

- `struct \_\_gnu\_parallel::balanced\_quicksort\_tag`
- `struct \_\_gnu\_parallel::balanced\_tag`
- `struct \_\_gnu\_parallel::constant\_size\_blocks\_tag`
- `struct \_\_gnu\_parallel::default\_parallel\_tag`
- `struct \_\_gnu\_parallel::equal\_split\_tag`
- `struct \_\_gnu\_parallel::exact\_tag`
- `struct \_\_gnu\_parallel::find\_tag`
- `struct \_\_gnu\_parallel::growing\_blocks\_tag`
- `struct \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag`
- `struct \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag`
- `struct \_\_gnu\_parallel::multiway\_mergesort\_tag`
- `struct \_\_gnu\_parallel::omp\_loop\_static\_tag`
- `struct \_\_gnu\_parallel::omp\_loop\_tag`
- `struct \_\_gnu\_parallel::parallel\_tag`
- `struct \_\_gnu\_parallel::quicksort\_tag`
- `struct \_\_gnu\_parallel::sampling\_tag`
- `struct \_\_gnu\_parallel::sequential\_tag`
- `struct \_\_gnu\_parallel::unbalanced\_tag`

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.638.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

## 6.639 types.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Typedefs

- typedef int64\_t [\\_\\_gnu\\_parallel::\\_CASable](#)
- typedef uint64\_t [\\_\\_gnu\\_parallel::\\_SequenceIndex](#)
- typedef uint16\_t [\\_\\_gnu\\_parallel::\\_ThreadIndex](#)

### Enumerations

- enum [\\_\\_gnu\\_parallel::\\_AlgorithmStrategy](#) { **heuristic** , **force\_sequential** , **force\_parallel** }
- enum [\\_\\_gnu\\_parallel::\\_FindAlgorithm](#) { **GROWING\_BLOCKS** , **CONSTANT\_SIZE\_BLOCKS** , **EQUAL\_SPLIT** }
- enum [\\_\\_gnu\\_parallel::\\_MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_\\_gnu\\_parallel::\\_Parallelism](#) { [\\_\\_gnu\\_parallel::sequential](#) , [\\_\\_gnu\\_parallel::parallel\\_unbalanced](#) , [\\_\\_gnu\\_parallel::parallel\\_balanced](#) , [\\_\\_gnu\\_parallel::parallel\\_omp\\_loop](#) , [\\_\\_gnu\\_parallel::parallel\\_omp\\_loop\\_static](#) , [\\_\\_gnu\\_parallel::parallel\\_taskqueue](#) }
- enum [\\_\\_gnu\\_parallel::\\_PartialSumAlgorithm](#) { **RECURSIVE** , **LINEAR** }
- enum [\\_\\_gnu\\_parallel::\\_SortAlgorithm](#) { **MWMS** , **QS** , **QS\_BALANCED** }
- enum [\\_\\_gnu\\_parallel::\\_SplittingAlgorithm](#) { **SAMPLING** , **EXACT** }

### Variables

- static const int [\\_\\_gnu\\_parallel::\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_\\_gnu\\_parallel::\\_CASable\\_mask](#)

### 6.639.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

## 6.640 unique\_copy.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename [\\_Iter](#), class [\\_OutputIterator](#)>  
[\\_OutputIterator](#) [\\_\\_gnu\\_parallel::parallel\\_unique\\_copy](#) ([\\_Iter](#) \_\_first, [\\_Iter](#) \_\_last, [\\_OutputIterator](#) \_\_result)
- template<typename [\\_Iter](#), class [\\_OutputIterator](#), class [\\_BinaryPredicate](#)>  
[\\_OutputIterator](#) [\\_\\_gnu\\_parallel::parallel\\_unique\\_copy](#) ([\\_Iter](#) \_\_first, [\\_Iter](#) \_\_last, [\\_OutputIterator](#) \_\_result, [\\_BinaryPredicate](#) \_\_binary\_pred)

### 6.640.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

## 6.641 workstealing.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_Job<\\_DifferenceTp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define _GLIBCXX_JOB_VOLATILE`

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
_Op __gnu_parallel::__for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _↵  
_Op __op, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator\_traits<_RAIter>::↵  
::difference_type __bound)`

### 6.641.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing.

Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. Journal of the ACM, 46(5):720-748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

## 6.642 print File Reference

### Macros

- `#define __glibcxx_want_print`
- `#define _GLIBCXX_PRINT`

### 6.642.1 Detailed Description

This is a Standard C++ Library header.

## 6.643 queue File Reference

### Macros

- `#define __glibcxx_want_adaptor_iterator_pair_constructor`
- `#define __glibcxx_want_containers_ranges`
- `#define _GLIBCXX_QUEUE`

### 6.643.1 Detailed Description

This is a Standard C++ Library header.

## 6.644 experimental/random File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_randint`
- `#define _GLIBCXX_EXPERIMENTAL_RANDOM`

### Functions

- [std::default\\_random\\_engine](#) & [std::experimental::\\_S\\_randint\\_engine](#) ()
- `template<typename _IntType>`  
`_IntType std::experimental::randint (_IntType __a, _IntType __b)`
- `void std::experimental::reseed ()`
- `void std::experimental::reseed (default_random_engine::result_type __value)`

#### 6.644.1 Detailed Description

This is a TS C++ Library header.

## 6.645 random File Reference

### Macros

- `#define _GLIBCXX_RANDOM`

#### 6.645.1 Detailed Description

This is a Standard C++ Library header.

## 6.646 ranges File Reference

### Macros

- `#define _GLIBCXX_RANGES`

#### 6.646.1 Detailed Description

This is a Standard C++ Library header.

## 6.647 experimental/ratio File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_RATIO`

## Variables

- `template<typename _R1, typename _R2>`  
`constexpr bool std::experimental::ratio_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool std::experimental::ratio_greater_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool std::experimental::ratio_greater_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool std::experimental::ratio_less_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool std::experimental::ratio_less_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool std::experimental::ratio_not_equal_v`

### 6.647.1 Detailed Description

This is a TS C++ Library header.

## 6.648 ratio File Reference

### Classes

- struct [`std::ratio< \_Num, \_Den >`](#)
- struct [`std::ratio\_equal< \_R1, \_R2 >`](#)
- struct [`std::ratio\_greater< \_R1, \_R2 >`](#)
- struct [`std::ratio\_greater\_equal< \_R1, \_R2 >`](#)
- struct [`std::ratio\_less< \_R1, \_R2 >`](#)
- struct [`std::ratio\_less\_equal< \_R1, \_R2 >`](#)
- struct [`std::ratio\_not\_equal< \_R1, \_R2 >`](#)

### Namespaces

- namespace [`std`](#)

### Macros

- `#define __glibcxx_want_ratio`
- `#define _GLIBCXX_RATIO`

### Typedefs

- using **std::atto**
- using **std::centi**
- using **std::deca**
- using **std::deci**
- using **std::exa**
- using **std::femto**
- using **std::giga**
- using **std::hecto**
- using **std::kilo**
- using **std::mega**
- using **std::micro**
- using **std::milli**



- using **std::nano**
- using **std::peta**
- using **std::pico**
- template<typename \_R1, typename \_R2>  
using [std::ratio\\_add](#)
- template<typename \_R1, typename \_R2>  
using [std::ratio\\_divide](#)
- template<typename \_R1, typename \_R2>  
using [std::ratio\\_multiply](#)
- template<typename \_R1, typename \_R2>  
using [std::ratio\\_subtract](#)
- using **std::tera**

### Variables

- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_equal\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_greater\_equal\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_greater\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_less\_equal\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_less\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_not\_equal\_v**

#### 6.648.1 Detailed Description

This is a Standard C++ Library header.

### 6.649 tr2/ratio File Reference

#### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

#### 6.649.1 Detailed Description

This is a TR2 C++ Library header.

### 6.650 experimental/regex File Reference

#### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

#### Macros

- **#define \_GLIBCXX\_EXPERIMENTAL\_REGEX**

## Typedefs

- typedef [match\\_results](#)< const char \* > **std::experimental::fundamentals\_v2::pmr::cmatch**
- template<typename [\\_BidirectionalIterator](#)>  
using **std::experimental::fundamentals\_v2::pmr::match\_results**
- typedef [match\\_results](#)< string::const\_iterator > **std::experimental::fundamentals\_v2::pmr::smatch**
- typedef [match\\_results](#)< const wchar\_t \* > **std::experimental::fundamentals\_v2::pmr::wcmatch**
- typedef [match\\_results](#)< wstring::const\_iterator > **std::experimental::fundamentals\_v2::pmr::wsmatch**

### 6.650.1 Detailed Description

This is a TS C++ Library header.

## 6.651 regex File Reference

### Namespaces

- namespace [std](#)

### Macros

- #define [\\_\\_glibcxx\\_want\\_nonmember\\_container\\_access](#)
- #define [\\_GLIBCXX\\_REGEX](#)

## Typedefs

- using **std::pmr::cmatch**
- template<typename [\\_BidirectionalIterator](#)>  
using **std::pmr::match\_results**
- using **std::pmr::smatch**
- using **std::pmr::wcmatch**
- using **std::pmr::wsmatch**

### 6.651.1 Detailed Description

This is a Standard C++ Library header.

## 6.652 scoped\_allocator File Reference

### Classes

- class [std::scoped\\_allocator\\_adaptor](#)< [\\_OuterAlloc](#), [\\_InnerAllocs](#) >

### Namespaces

- namespace [std](#)

### Macros

- #define [\\_\\_glibcxx\\_want\\_allocator\\_traits\\_is\\_always\\_equal](#)
- #define [\\_SCOPED\\_ALLOCATOR](#)

### 6.652.1 Detailed Description

This is a Standard C++ Library header.

## 6.653 semaphore File Reference

### Macros

- `#define __glibcxx_want_semaphore`
- `#define _GLIBCXX_SEMAPHORE`

### 6.653.1 Detailed Description

This is a Standard C++ Library header.

## 6.654 debug/set File Reference

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Macros

- `#define _GLIBCXX_DEBUG_SET`

### 6.654.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.655 experimental/set File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_SET`

### Typedefs

- `template<typename _Key, typename _Compare = less<_Key>>>`  
using `std::experimental::fundamentals_v2::pmr::multiset`
- `template<typename _Key, typename _Compare = less<_Key>>>`  
using `std::experimental::fundamentals_v2::pmr::set`

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
void `std::experimental::erase_if` (`multiset`< \_Key, \_Compare, \_Alloc > &\_\_cont, \_Predicate \_\_pred)
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
void `std::experimental::erase_if` (`set`< \_Key, \_Compare, \_Alloc > &\_\_cont, \_Predicate \_\_pred)

### 6.655.1 Detailed Description

This is a TS C++ Library header.

## 6.656 set File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_containers_ranges`
- `#define __glibcxx_want_erase_if`
- `#define __glibcxx_want_generic_associative_lookup`
- `#define __glibcxx_want_node_extract`
- `#define __glibcxx_want_nonmember_container_access`
- `#define _GLIBCXX_SET`

### Typedefs

- `template<typename _Key, typename _Cmp = std::less<_Key>>  
using std::pmr::multiset`
- `template<typename _Key, typename _Cmp = std::less<_Key>>  
using std::pmr::set`

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>  
multiset< _Key, _Compare, _Alloc >::size_type std::erase\_if (multiset< _Key, _Compare, _Alloc > &__cont,  
_Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>  
set< _Key, _Compare, _Alloc >::size_type std::erase\_if (set< _Key, _Compare, _Alloc > &__cont, _Predicate  
__pred)`

#### 6.656.1 Detailed Description

This is a Standard C++ Library header.

## 6.657 shared\_mutex File Reference

### Classes

- class [std::shared\\_lock<\\_Mutex>](#)
- class [std::shared\\_timed\\_mutex](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_shared_mutex`
- `#define __glibcxx_want_shared_timed_mutex`
- `#define _GLIBCXX_SHARED_MUTEX`

#### 6.657.1 Detailed Description

This is a Standard C++ Library header.

## 6.658 `source_location` File Reference

### Macros

- `#define __glibcxx_want_source_location`
- `#define _GLIBCXX_SRCLOC`

#### 6.658.1 Detailed Description

This is a Standard C++ Library header.

## 6.659 `span` File Reference

### Macros

- `#define __glibcxx_want_span`
- `#define __glibcxx_want_span_initializer_list`
- `#define _GLIBCXX_SPAN`

#### 6.659.1 Detailed Description

This is a Standard C++ Library header.

## 6.660 `sstream` File Reference

### Classes

- class `std::basic_istream<_CharT, _Traits, _Alloc>`
- class `std::basic_ostringstream<_CharT, _Traits, _Alloc>`
- class `std::basic_stringbuf<_CharT, _Traits, _Alloc>`
- class `std::basic_stringstream<_CharT, _Traits, _Alloc>`

### Namespaces

- namespace `std`

### Macros

- `#define __glibcxx_want_sstream_from_string_view`
- `#define _GLIBCXX_LVAL_REF_QUAL`
- `#define _GLIBCXX_SSTREAM`
- `#define _GLIBCXX_SSTREAM_ALWAYS_INLINE`

### Functions

- `template<class _CharT, class _Traits, class _Allocator>`  
`void std::swap (basic_istream<_CharT, _Traits, _Allocator> &__x, basic_istream<_CharT, _Traits, _Allocator> &__y)`
- `template<class _CharT, class _Traits, class _Allocator>`  
`void std::swap (basic_ostringstream<_CharT, _Traits, _Allocator> &__x, basic_ostringstream<_CharT, _Traits, _Allocator> &__y)`
- `template<class _CharT, class _Traits, class _Allocator>`  
`void std::swap (basic_stringbuf<_CharT, _Traits, _Allocator> &__x, basic_stringbuf<_CharT, _Traits, _Allocator> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _CharT, class _Traits, class _Allocator>`  
`void std::swap (basic_stringstream<_CharT, _Traits, _Allocator> &__x, basic_stringstream<_CharT, _Traits, _Allocator> &__y)`

### 6.660.1 Detailed Description

This is a Standard C++ Library header.

## 6.661 stack File Reference

### Macros

- `#define __glibcxx_want_adaptor_iterator_pair_constructor`
- `#define __glibcxx_want_containers_ranges`
- `#define _GLIBCXX_STACK`

### 6.661.1 Detailed Description

This is a Standard C++ Library header.

## 6.662 stdatomic.h File Reference

### 6.662.1 Detailed Description

This is a Standard C++ Library header.

## 6.663 stdbit.h File Reference

### 6.663.1 Detailed Description

This is a Standard C++ Library header.

## 6.664 stdckdint.h File Reference

### 6.664.1 Detailed Description

This is a Standard C++ Library header.

## 6.665 stdexcept File Reference

### Classes

- class `std::domain_error`
- class `std::invalid_argument`
- class `std::length_error`
- class `std::logic_error`
- class `std::out_of_range`
- class `std::overflow_error`
- class `std::range_error`
- class `std::runtime_error`
- class `std::underflow_error`

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_STDEXCEPT`

## Typedefs

- typedef [basic\\_string](#)< char > **std::\_\_sso\_string**

### 6.665.1 Detailed Description

This is a Standard C++ Library header.

## 6.666 stdlib.h File Reference

### Functions

- template<typename [\\_Tp](#)>  
  [\\_Tp](#) [abs](#) (const complex< [\\_Tp](#) > &)
- [ldiv\\_t](#) [div](#) (long [\\_\\_i](#), long [\\_\\_j](#)) noexcept

### 6.666.1 Detailed Description

This is a Standard C++ Library header.

### 6.666.2 Function Documentation

#### **abs()**

```
template<typename \_Tp>
\_Tp std::abs (
    const complex< \_Tp > & \_\_z) [inline]
```

Return magnitude of [z](#).

## 6.667 stop\_token File Reference

### 6.667.1 Detailed Description

This is a Standard C++ Library header.

## 6.668 streambuf File Reference

### Classes

- class **std::basic\_streambuf**< [\\_CharT](#), [\\_Traits](#) >

### Namespaces

- namespace [std](#)

### Macros

- #define **\_GLIBXX\_STREAMBUF**
- #define **\_IsUnused**

### Functions

- template<typename [\\_CharT](#), typename [\\_Traits](#)>  
  [streamsize](#) **std::\_\_copy\_streambufs\_eof** ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_sbin](#), [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_sbout](#), bool &[\\_\\_ineof](#))
- template<> [streamsize](#) **std::\_\_copy\_streambufs\_eof** ([basic\\_streambuf](#)< char > \*[\\_\\_sbin](#), [basic\\_streambuf](#)< char > \*[\\_\\_sbout](#), bool &[\\_\\_ineof](#))

- template<> [streamsize](#) [std::\\_\\_copy\\_streambufs\\_eof](#) ([basic\\_streambuf](#)< [wchar\\_t](#)> \*[\\_\\_sbin](#), [basic\\_streambuf](#)< [wchar\\_t](#)> \*[\\_\\_sbout](#), [bool](#) &[\\_\\_ineof](#))

### 6.668.1 Detailed Description

This is a Standard C++ Library header.

## 6.669 debug/string File Reference

### Classes

- class [\\_\\_gnu\\_debug::basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_Allocator](#) >
- struct [std::hash](#)< [\\_\\_gnu\\_debug::basic\\_string](#)< [\\_CharT](#) > >

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)

### Macros

- #define [\\_\\_glibcxx\\_check\\_string](#)([\\_String](#))
- #define [\\_\\_glibcxx\\_check\\_string\\_constructor](#)([\\_Str](#))
- #define [\\_\\_glibcxx\\_check\\_string\\_len](#)([\\_String](#), [\\_Len](#))
- #define [\\_\\_glibcxx\\_check\\_string\\_n\\_constructor](#)([\\_Str](#), [\\_Size](#))
- #define [\\_GLIBCXX\\_DEBUG\\_STRING](#)
- #define [\\_GLIBCXX\\_DEBUG\\_VERIFY\\_STR\\_COND\\_AT](#)([\\_Cond](#), [\\_File](#), [\\_Line](#), [\\_Func](#))
- #define [\\_GLIBCXX\\_INSERT\\_RETURNS\\_ITERATOR](#)
- #define [\\_GLIBCXX\\_INSERT\\_RETURNS\\_ITERATOR\\_ONLY](#)([expr](#))

### Typedefs

- typedef [basic\\_string](#)< [char](#) > [\\_\\_gnu\\_debug::string](#)
- typedef [basic\\_string](#)< [char16\\_t](#) > [\\_\\_gnu\\_debug::u16string](#)
- typedef [basic\\_string](#)< [char32\\_t](#) > [\\_\\_gnu\\_debug::u32string](#)
- typedef [basic\\_string](#)< [wchar\\_t](#) > [\\_\\_gnu\\_debug::wstring](#)

### Functions

- template<typename [\\_CharT](#), typename [\\_Integer](#)>  
const [\\_CharT](#) \* [\\_\\_gnu\\_debug::\\_\\_check\\_string](#) (const [\\_CharT](#) \*[\\_\\_s](#), [\\_Integer](#) [\\_\\_n](#), const [char](#) \*[\\_\\_file](#), unsigned int [\\_\\_line](#), const [char](#) \*[\\_\\_function](#))
- template<typename [\\_CharT](#)>  
const [\\_CharT](#) \* [\\_\\_gnu\\_debug::\\_\\_check\\_string](#) (const [\\_CharT](#) \*[\\_\\_s](#), const [char](#) \*[\\_\\_file](#), unsigned int [\\_\\_line](#), const [char](#) \*[\\_\\_function](#))
- template<typename [\\_CharT](#), typename [\\_Traits](#), typename [\\_Allocator](#)>  
[std::basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > & [\\_\\_gnu\\_debug::getline](#) ([std::basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > &[\\_\\_is](#), [basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_Allocator](#) > &[\\_\\_str](#))
- template<typename [\\_CharT](#), typename [\\_Traits](#), typename [\\_Allocator](#)>  
[std::basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > & [\\_\\_gnu\\_debug::getline](#) ([std::basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > &[\\_\\_is](#), [basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_Allocator](#) > &[\\_\\_str](#), [\\_CharT](#) [\\_\\_delim](#))
- template<typename [\\_CharT](#), typename [\\_Traits](#), typename [\\_Allocator](#)>  
[bool](#) [\\_\\_gnu\\_debug::operator!=](#) (const [\\_CharT](#) \*[\\_\\_lhs](#), const [basic\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_Allocator](#) > &[\\_\\_rhs](#))



- [illegible]

- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator> (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__↵`  
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator> (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string<`  
`_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator>= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__↵`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__↵`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string<`  
`_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic\_istream< _CharT, _Traits > & __gnu_debug::operator>> (std::basic\_istream< _CharT, _Traits >`  
`&__is, basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`void __gnu_debug::swap (basic\_string< _CharT, _Traits, _Allocator > &__lhs, basic\_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`

### 6.669.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.670 experimental/string File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_STRING`

### Typedefs

- `template<typename _CharT, typename _Traits = char_traits<_CharT>>`  
`using std::experimental::fundamentals\_v2::pmr::basic\_string`
- `typedef basic\_string< char > std::experimental::fundamentals\_v2::pmr::string`
- `typedef basic\_string< char16_t > std::experimental::fundamentals\_v2::pmr::u16string`
- `typedef basic\_string< char32_t > std::experimental::fundamentals\_v2::pmr::u32string`
- `typedef basic\_string< wchar_t > std::experimental::fundamentals\_v2::pmr::wstring`

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Up>`  
`void std::experimental::erase (basic\_string< _CharT, _Traits, _Alloc > &__cont, const _Up &__value)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase\_if (basic\_string< _CharT, _Traits, _Alloc > &__cont, _Predicate __pred)`

### 6.670.1 Detailed Description

This is a TS C++ Library header.

## 6.671 string File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_algorithm_default_value_type`
- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_constexpr_char_traits`
- `#define __glibcxx_want_constexpr_string`
- `#define __glibcxx_want_containers_ranges`
- `#define __glibcxx_want_erase_if`
- `#define __glibcxx_want_nonmember_container_access`
- `#define __glibcxx_want_string_resize_and_overwrite`
- `#define __glibcxx_want_string_udls`
- `#define __glibcxx_want_to_string`
- `#define _GLIBCXX_STRING`

### Typedefs

- `template<typename _CharT, typename _Traits = char_traits<_CharT>>>  
using std::pmr::basic_string`
- `using std::pmr::string`
- `using std::pmr::u16string`
- `using std::pmr::u32string`
- `using std::pmr::wstring`

### 6.671.1 Detailed Description

This is a Standard C++ Library header.

## 6.672 experimental/string\_view File Reference

### Classes

- class [std::experimental::fundamentals\\_v1::basic\\_string\\_view< \\_CharT, \\_Traits >](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_string_view`
- `#define _GLIBCXX_EXPERIMENTAL_STRING_VIEW`

## Typedefs

- using **std::experimental::string\_view**
- using **std::experimental::u16string\_view**
- using **std::experimental::u32string\_view**
- using **std::experimental::wstring\_view**

## Functions

- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator!=** ([\\_\\_type\\_identity\\_t](#)< [basic\\_string\\_view](#)< \_CharT, \_Traits > > <←  
\_\_x, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator!=** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [\\_\\_type\\_identity\\_t](#)<  
[basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator!=** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [basic\\_string\\_view](#)<  
\_CharT, \_Traits > \_\_y) noexcept
- constexpr [basic\\_string\\_view](#)< char > **std::experimental::literals::operator""sv** (const char \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< char16\_t > **std::experimental::literals::operator""sv** (const char16\_t \*\_\_str,  
size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< char32\_t > **std::experimental::literals::operator""sv** (const char32\_t \*\_\_str,  
size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< wchar\_t > **std::experimental::literals::operator""sv** (const wchar\_t \*\_\_str,  
size\_t \_\_len) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator**< ([\\_\\_type\\_identity\\_t](#)< [basic\\_string\\_view](#)< \_CharT, \_Traits > > <←  
\_\_x, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator**< ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [\\_\\_type\\_identity\\_t](#)<  
[basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator**< ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [basic\\_string\\_view](#)<  
\_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **std::experimental::operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &<←  
\_\_os, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_str)
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator**<= ([\\_\\_type\\_identity\\_t](#)< [basic\\_string\\_view](#)< \_CharT, \_Traits > > <  
\_\_x, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator**<= ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [\\_\\_type\\_identity\\_t](#)<  
[basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator**<= ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [basic\\_string\\_view](#)<  
\_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator**== ([\\_\\_type\\_identity\\_t](#)< [basic\\_string\\_view](#)< \_CharT, \_Traits > > <←  
\_\_x, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator**== ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [\\_\\_type\\_identity\\_t](#)<  
[basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept

- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator== (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator> ( (__type_identity_t< basic_string_view< _CharT, _Traits > > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator> (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator> (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator>= ( (__type_identity_t< basic_string_view< _CharT, _Traits > > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator>= (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator>= (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`

### 6.672.1 Detailed Description

This is a TS C++ Library header.

## 6.673 string\_view File Reference

### Classes

- class `std::basic_string_view< _CharT, _Traits >`

### Namespaces

- namespace `std`
- namespace `std::literals`

### Macros

- `#define __glibcxx_want_constexpr_char_traits`
- `#define __glibcxx_want_constexpr_string_view`
- `#define __glibcxx_want_freestanding_string_view`
- `#define __glibcxx_want_starts_ends_with`
- `#define __glibcxx_want_string_contains`
- `#define __glibcxx_want_string_view`
- `#define _GLIBCXX_STRING_VIEW`

### Typedefs

- using `std::string_view`
- using `std::u16string_view`
- using `std::u32string_view`
- using `std::wstring_view`

## Functions

- constexpr size\_t **std::\_\_sv\_check** (size\_t \_\_size, size\_t \_\_pos, const char \*\_\_s)
- constexpr size\_t **std::\_\_sv\_limit** (size\_t \_\_size, size\_t \_\_pos, size\_t \_\_off) noexcept
- constexpr [basic\\_string\\_view](#)< char > **std::literals::operator""sv** (const char \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< char16\_t > **std::literals::operator""sv** (const char16\_t \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< char32\_t > **std::literals::operator""sv** (const char32\_t \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< wchar\_t > **std::literals::operator""sv** (const wchar\_t \*\_\_str, size\_t \_\_len) noexcept
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **std::operator<<** ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_str)
- template<typename \_CharT, typename \_Traits>  
constexpr auto **std::operator<=>** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, \_\_type\_identity\_t< [basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept -> decltype(\_\_detail::\_\_char\_traits\_cmp\_cat< \_Traits >(0))
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::operator==** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, \_\_type\_identity\_t< [basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept

### 6.673.1 Detailed Description

This is a Standard C++ Library header.

## 6.674 syncstream File Reference

### Macros

- #define **\_\_glibcxx\_want\_syncbuf**
- #define **\_GLIBCXX\_SYNCSTREAM**

### 6.674.1 Detailed Description

This is a Standard C++ Library header.

## 6.675 experimental/system\_error File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- #define **\_GLIBCXX\_EXPERIMENTAL\_SYSTEM\_ERROR**

### Variables

- template<typename \_Tp>  
constexpr bool **std::experimental::is\_error\_code\_enum\_v**
- template<typename \_Tp>  
constexpr bool **std::experimental::is\_error\_condition\_enum\_v**

### 6.675.1 Detailed Description

This is a TS C++ Library header.

## 6.676 `system_error` File Reference

### Classes

- class [std::error\\_category](#)
- class [std::error\\_code](#)
- class [std::error\\_condition](#)
- struct [std::hash< error\\_code >](#)
- struct [std::hash< error\\_condition >](#)
- struct [std::is\\_error\\_code\\_enum< \\_Tp >](#)
- struct [std::is\\_error\\_condition\\_enum< \\_Tp >](#)
- class [std::system\\_error](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_SYSTEM_ERROR`

### Functions

- const [error\\_category](#) & [std::generic\\_category](#) () noexcept
- void [std::\\_\\_adl\\_only::make\\_error\\_code](#) ()=delete
- void [std::\\_\\_adl\\_only::make\\_error\\_condition](#) ()=delete
- const [error\\_category](#) & [std::system\\_category](#) () noexcept

### Variables

- `template<typename _Tp>`  
`constexpr bool std::is\_error\_code\_enum\_v`
- `template<typename _Tp>`  
`constexpr bool std::is\_error\_condition\_enum\_v`

### 6.676.1 Detailed Description

This is a Standard C++ Library header.

## 6.677 `text_encoding` File Reference

### 6.677.1 Detailed Description

This is a Standard C++ Library header.

## 6.678 `tgmath.h` File Reference

### 6.678.1 Detailed Description

This is a Standard C++ Library header.

## 6.679 thread File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_formatters`
- `#define __glibcxx_want_jthread`
- `#define _GLIBCXX_THREAD`

### 6.679.1 Detailed Description

This is a Standard C++ Library header.

## 6.680 bool\_set File Reference

### Classes

- class [std::tr2::bool\\_set](#)

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

### Macros

- `#define _GLIBCXX_TR2_BOOL_SET`

### Functions

- `bool std::tr2::certainly (bool_set __b)`
- `bool std::tr2::contains (bool_set __s, bool_set __t)`
- `bool std::tr2::equals (bool_set __s, bool_set __t)`
- `bool std::tr2::is_emptyset (bool_set __b)`
- `bool std::tr2::is_indeterminate (bool_set __b)`
- `bool std::tr2::is_singleton (bool_set __b)`
- `bool_set std::tr2::operator!= (bool __s, bool_set __t)`
- `bool_set std::tr2::operator!= (bool_set __s, bool __t)`
- `bool_set std::tr2::operator!= (bool_set __s, bool_set __t)`
- `bool_set std::tr2::operator& (bool __s, bool_set __t)`
- `bool_set std::tr2::operator& (bool_set __s, bool __t)`
- `bool_set std::tr2::operator== (bool __s, bool_set __t)`
- `bool_set std::tr2::operator== (bool_set __s, bool __t)`
- `bool_set std::tr2::operator^ (bool __s, bool_set __t)`
- `bool_set std::tr2::operator^ (bool_set __s, bool __t)`
- `bool_set std::tr2::operator| (bool __s, bool_set __t)`
- `bool_set std::tr2::operator| (bool_set __s, bool __t)`
- `bool std::tr2::possibly (bool_set __b)`
- `bool_set std::tr2::set_complement (bool_set __b)`
- `bool_set std::tr2::set_intersection (bool __s, bool_set __t)`
- `bool_set std::tr2::set_intersection (bool_set __s, bool __t)`



- [bool\\_set std::tr2::set\\_intersection](#) ([bool\\_set \\_\\_s](#), [bool\\_set \\_\\_t](#))
- [bool\\_set std::tr2::set\\_union](#) ([bool \\_\\_s](#), [bool\\_set \\_\\_t](#))
- [bool\\_set std::tr2::set\\_union](#) ([bool\\_set \\_\\_s](#), [bool \\_\\_t](#))
- [bool\\_set std::tr2::set\\_union](#) ([bool\\_set \\_\\_s](#), [bool\\_set \\_\\_t](#))

### 6.680.1 Detailed Description

This is a TR2 C++ Library header.

## 6.681 bool\_set.tcc File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

### Macros

- `#define _GLIBCXX_TR2_BOOL_SET_TCC`

### 6.681.1 Detailed Description

This is a TR2 C++ Library header.

## 6.682 dynamic\_bitset File Reference

### Classes

- struct [std::tr2::\\_\\_dynamic\\_bitset\\_base](#)< [\\_WordT](#), [\\_Alloc](#) >
- class [std::tr2::dynamic\\_bitset](#)< [\\_WordT](#), [\\_Alloc](#) >
- class [std::tr2::dynamic\\_bitset](#)< [\\_WordT](#), [\\_Alloc](#) >::reference

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

### Macros

- `#define _GLIBCXX_TR2_DYNAMIC_BITSET`

### Functions

- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>  
std::basic\_ostream< \_CharT, \_Traits > & std::tr2::operator<< (std::basic\_ostream< \_CharT, \_Traits > &__os,  
const dynamic\_bitset< \_WordT, \_Alloc > &__x)`
- `template<typename _WordT, typename _Alloc>  
bool std::tr2::operator!= (const dynamic\_bitset< \_WordT, \_Alloc > &__lhs, const dynamic\_bitset< \_WordT, \_Alloc  
> &__rhs)`
- `template<typename _WordT, typename _Alloc>  
bool std::tr2::operator<= (const dynamic\_bitset< \_WordT, \_Alloc > &__lhs, const dynamic\_bitset< \_WordT, \_Alloc > &__rhs)`

- `template<typename _WordT, typename _Alloc>`  
`bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>`  
`bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`

### 6.682.1 Detailed Description

This is a TR2 C++ Library header.

## 6.683 dynamic\_bitset.tcc File Reference

### Namespaces

- namespace `std`
- namespace `std::tr2`

### Macros

- `#define _GLIBCXX_TR2_DYNAMIC_BITSET_TCC`

### Functions

- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>`  
`std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (std::basic_istream< _CharT, _Traits > &__is, dynamic_bitset< _WordT, _Alloc > &__x)`

### 6.683.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<tr2/dynamic_bitset>`.

## 6.684 experimental/tuple File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

## Macros

- `#define __cpp_lib_experimental_tuple`
- `#define _GLIBCXX_EXPERIMENTAL_TUPLE`

## Functions

- `template<typename _Fn, typename _Tuple, std::size_t... _Idx>`  
`constexpr decltype(auto) std::experimental::__apply_impl (_Fn &&__f, _Tuple &&__t, std::index\_sequence<  
_Idx... >)`
- `template<typename _Fn, typename _Tuple>`  
`constexpr decltype(auto) std::experimental::apply (_Fn &&__f, _Tuple &&__t)`

## Variables

- `template<typename _Tp>`  
`constexpr size_t std::experimental::tuple_size_v`

### 6.684.1 Detailed Description

This is a TS C++ Library header.

## 6.685 tuple File Reference

### Classes

- class [std::tuple< \\_Elements >](#)
- class [std::tuple< \\_T1, \\_T2 >](#)
- struct [std::tuple\\_element< \\_\\_i, tuple< \\_Types... > >](#)
- struct [std::tuple\\_size< tuple< \\_Elements... > >](#)
- struct [std::uses\\_allocator< tuple< \\_Types... >, \\_Alloc >](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_apply`
- `#define __glibcxx_want_constexpr_tuple`
- `#define __glibcxx_want_constrained_equality`
- `#define __glibcxx_want_make_from_tuple`
- `#define __glibcxx_want_ranges_zip`
- `#define __glibcxx_want_tuple_element_t`
- `#define __glibcxx_want_tuple_like`
- `#define __glibcxx_want_tuples_by_type`
- `#define _GLIBCXX_TUPLE`

### Functions

- `template<size_t __i, typename _Head, typename... _Tail>`  
`constexpr _Head & std::__get_helper (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<size_t __i, typename _Head, typename... _Tail>`  
`constexpr const _Head & std::__get_helper (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`

- `template<size_t __i, typename... _Types>  
__enable_if_t<(__i >= sizeof...(_Types))> std::__get_helper (const tuple< _Types... > &)=delete`
- `template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>  
constexpr _Cat std::__tuple_cmp (const _Tp &__t, const _Up &__u, _IndexSeq __indices)`
- `template<typename... _Elements>  
constexpr tuple< _Elements &&... > std::forward_as_tuple (_Elements &&... __args) noexcept`
- `template<size_t __i, typename... _Elements>  
constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept`
- `template<size_t __i, typename... _Elements>  
constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept`
- `template<size_t __i, typename... _Elements>  
constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept`
- `template<size_t __i, typename... _Elements>  
constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept`
- `template<typename... _Elements>  
constexpr tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (_Elements &&... __args)`
- `template<typename... _Tps, typename... _Ups>  
requires (sizeof...(_Tps) == sizeof...(_Ups)) && (requires { typename __detail::__synth3way_t<_Tps, _Ups>; } && ...)  
constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > std::operator<=> (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`
- `template<typename... _Tps, typename... _Ups>  
requires (sizeof...(_Tps) == sizeof...(_Ups)) && (requires (const _Tps& __t, const _Ups& __u) { { __t == __u } -> __detail::__boolean_testable; } && ...)  
constexpr bool std::operator== (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`
- `template<typename... _Elements>  
constexpr enable_if<!__and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &, tuple< _Elements... > &)=delete`
- `template<typename... _Elements>  
constexpr enable_if< __and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>  
constexpr tuple< _Elements &&... > std::tie (_Elements &&... __args) noexcept`
- `template<typename... _UTypes>  
std::tuple (_UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename... _UTypes>  
std::tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename _T1, typename _T2>  
std::tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >`
- `template<typename _Alloc, typename... _UTypes>  
std::tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >`
- `template<typename _T1, typename _T2>  
std::tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >`
- `template<typename... _Tpls, typename = typename enable_if<__and< __is_tuple_like<_Tpls>...>::value>::type>  
constexpr auto std::tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type`

## Variables

- `template<typename... _Types>  
constexpr size_t std::tuple_size_v< const tuple< _Types... > >`

- `template<typename... _Types>`  
`constexpr size_t std::tuple_size_v< tuple< _Types... > >`

### 6.685.1 Detailed Description

This is a Standard C++ Library header.

## 6.686 `experimental/type_traits` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_TYPE_TRAITS`
- `#define __cpp_lib_experimental_type_trait_variable_templates`
- `template<typename _Tp>`  
`constexpr size_t std::experimental::alignment_of_v`
- `template<typename _Tp, unsigned _Idx = 0>`  
`constexpr size_t std::experimental::extent_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::has_virtual_destructor_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_abstract_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_arithmetic_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_array_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_assignable_v`
- `template<typename _Base, typename _Derived>`  
`constexpr bool std::experimental::is_base_of_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_class_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_compound_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_const_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_constructible_v`
- `template<typename _From, typename _To>`  
`constexpr bool std::experimental::is_convertible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_destructible_v`

- `template<typename _Tp>`  
`constexpr bool std::experimental::is_empty_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_enum_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_final_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_floating_point_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_function_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_fundamental_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_integral_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_literal_type_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_lvalue_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_function_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_object_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_nothrow_assignable_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_nothrow_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_destructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_move_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_null_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_object_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_pod_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_pointer_v`

- `template<typename _Tp>`  
`constexpr bool std::experimental::is_polymorphic_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_rvalue_reference_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_same_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_same_v< _Tp, _Tp >`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_scalar_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_signed_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_standard_layout_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivial_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_trivially_assignable_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_trivially_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copyable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_destructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_move_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_union_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_unsigned_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_void_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_volatile_v`
- `template<typename _Tp>`  
`constexpr size_t std::experimental::rank_v`
- `#define \_\_cpp\_lib\_experimental\_detect`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::detected\_or`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::detected\_or\_t`

- `template<template< typename... > class _Op, typename... _Args>`  
using [std::experimental::detected\\_t](#)
- `template<template< typename... > class _Op, typename... _Args>`  
using [std::experimental::is\\_detected](#)
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
using [std::experimental::is\\_detected\\_convertible](#)
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
constexpr bool [std::experimental::is\\_detected\\_convertible\\_v](#)
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
using [std::experimental::is\\_detected\\_exact](#)
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
constexpr bool [std::experimental::is\\_detected\\_exact\\_v](#)
- `template<template< typename... > class _Op, typename... _Args>`  
constexpr bool [std::experimental::is\\_detected\\_v](#)
- `template<typename...>`  
using [std::experimental::void\\_t](#)
- `#define __cpp_lib_experimental_logical_traits`
- `template<typename... _Bn>`  
constexpr bool [std::experimental::conjunction\\_v](#)
- `template<typename... _Bn>`  
constexpr bool [std::experimental::disjunction\\_v](#)
- `template<typename _Pp>`  
constexpr bool [std::experimental::negation\\_v](#)

### 6.686.1 Detailed Description

This is a TS C++ Library header.

This header defines variable templates for the C++14 type traits.

Equivalent variable templates are defined in namespace `std` since C++17.

See also

[variable\\_templates](#)

Since

C++14

## 6.687 tr2/type\_traits File Reference

### Classes

- struct [std::tr2::\\_\\_reflection\\_typelist< \\_First, \\_Rest... >](#)
- struct [std::tr2::\\_\\_reflection\\_typelist<>](#)
- struct [std::tr2::bases< \\_Tp >](#)
- struct [std::tr2::direct\\_bases< \\_Tp >](#)

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

### Macros

- `#define _GLIBCXX_TR2_TYPE_TRAITS`



### 6.687.1 Detailed Description

This is a TR2 C++ Library header.

## 6.688 `type_traits` File Reference

### Classes

- struct `std::add_const< _Tp >`
- struct `std::add_cv< _Tp >`
- struct `std::add_lvalue_reference< _Tp >`
- struct `std::add_pointer< _Tp >`
- struct `std::add_rvalue_reference< _Tp >`
- struct `std::add_volatile< _Tp >`
- struct `std::aligned_storage< _Len, _Align >`
- struct `std::aligned_union< _Len, _Types >`
- struct `std::alignment_of< _Tp >`
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
- struct `std::decay< _Tp >`
- struct `std::enable_if< bool, _Tp >`
- struct `std::extent< typename, _Uint >`
- struct `std::has_virtual_destructor< _Tp >`
- struct `std::integral_constant< _Tp, __v >`
- struct `std::is_abstract< _Tp >`
- struct `std::is_arithmetic< _Tp >`
- struct `std::is_array< _Tp >`
- struct `std::is_assignable< _Tp, _Up >`
- struct `std::is_base_of< _Base, _Derived >`
- struct `std::is_class< _Tp >`
- struct `std::is_compound< _Tp >`
- struct `std::is_const< _Tp >`
- struct `std::is_constructible< _Tp, _Args >`
- struct `std::is_copy_assignable< _Tp >`
- struct `std::is_copy_constructible< _Tp >`
- struct `std::is_default_constructible< _Tp >`
- struct `std::is_destructible< _Tp >`
- struct `std::is_empty< _Tp >`
- struct `std::is_enum< _Tp >`
- struct `std::is_floating_point< _Tp >`
- struct `std::is_function< _Tp >`
- struct `std::is_fundamental< _Tp >`
- struct `std::is_integral< _Tp >`
- struct `std::is_layout_compatible< _Tp, _Up >`
- struct `std::is_literal_type< _Tp >`
- struct `std::is_lvalue_reference< typename >`
- struct `std::is_member_function_pointer< _Tp >`
- struct `std::is_member_object_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_move_assignable< _Tp >`
- struct `std::is_move_constructible< _Tp >`
- struct `std::is_nothrow_assignable< _Tp, _Up >`
- struct `std::is_nothrow_constructible< _Tp, _Args >`

- struct `std::is_nothrow_copy_assignable< _Tp >`
- struct `std::is_nothrow_copy_constructible< _Tp >`
- struct `std::is_nothrow_default_constructible< _Tp >`
- struct `std::is_nothrow_destructible< _Tp >`
- struct `std::is_nothrow_move_assignable< _Tp >`
- struct `std::is_nothrow_move_constructible< _Tp >`
- struct `std::is_object< _Tp >`
- struct `std::is_pod< _Tp >`
- struct `std::is_pointer< _Tp >`
- struct `std::is_pointer_interconvertible_base_of< _Base, _Derived >`
- struct `std::is_polymorphic< _Tp >`
- struct `std::is_reference< _Tp >`
- struct `std::is_rvalue_reference< typename >`
- struct `std::is_same< _Tp, _Up >`
- struct `std::is_scalar< _Tp >`
- struct `std::is_signed< _Tp >`
- struct `std::is_standard_layout< _Tp >`
- struct `std::is_trivial< _Tp >`
- struct `std::is_trivially_assignable< _Tp, _Up >`
- struct `std::is_trivially_constructible< _Tp, _Args >`
- struct `std::is_trivially_copy_assignable< _Tp >`
- struct `std::is_trivially_copy_constructible< _Tp >`
- struct `std::is_trivially_copyable< _Tp >`
- struct `std::is_trivially_default_constructible< _Tp >`
- struct `std::is_trivially_destructible< _Tp >`
- struct `std::is_trivially_move_assignable< _Tp >`
- struct `std::is_trivially_move_constructible< _Tp >`
- struct `std::is_union< _Tp >`
- struct `std::is_unsigned< _Tp >`
- struct `std::is_void< _Tp >`
- struct `std::is_volatile< _Tp >`
- struct `std::make_signed< _Tp >`
- struct `std::make_unsigned< _Tp >`
- struct `std::rank< _Tp >`
- struct `std::remove_all_extents< _Tp >`
- struct `std::remove_const< _Tp >`
- struct `std::remove_cv< _Tp >`
- struct `std::remove_extent< _Tp >`
- struct `std::remove_pointer< _Tp >`
- struct `std::remove_reference< _Tp >`
- struct `std::remove_volatile< _Tp >`
- struct `std::underlying_type< _Tp >`

## Namespaces

- namespace `std`

## Macros

- `#define __glibcxx_want_bool_constant`
- `#define __glibcxx_want_bounded_array_traits`
- `#define __glibcxx_want_constant_wrapper`
- `#define __glibcxx_want_has_unique_object_representations`
- `#define __glibcxx_want_integral_constant_callable`
- `#define __glibcxx_want_is_aggregate`
- `#define __glibcxx_want_is_constant_evaluated`
- `#define __glibcxx_want_is_final`
- `#define __glibcxx_want_is_invocable`
- `#define __glibcxx_want_is_layout_compatible`
- `#define __glibcxx_want_is_nothrow_convertible`
- `#define __glibcxx_want_is_null_pointer`
- `#define __glibcxx_want_is_pointer_interconvertible`
- `#define __glibcxx_want_is_scoped_enum`
- `#define __glibcxx_want_is_swappable`
- `#define __glibcxx_want_is_virtual_base_of`
- `#define __glibcxx_want_logical_traits`
- `#define __glibcxx_want_reference_from_temporary`
- `#define __glibcxx_want_remove_cvref`
- `#define __glibcxx_want_result_of_sfinae`
- `#define __glibcxx_want_transformation_trait_aliases`
- `#define __glibcxx_want_type_identity`
- `#define __glibcxx_want_type_trait_variable_templates`
- `#define __glibcxx_want_unwrap_ref`
- `#define __glibcxx_want_void_t`
- `#define __GLIBCXX_TYPE_TRAITS`

## Typedefs

- `template<bool _Cond, typename _If, typename _Else>`  
`using std::__conditional_t`
- `template<bool _Cond, typename _Tp = void>`  
`using std::__enable_if_t`
- `template<typename _ToElementType, typename _FromElementType>`  
`using std::__is_array_convertible`
- `template<typename _Tp>`  
`using std::add_lvalue_reference_t`
- `template<typename _Tp>`  
`using std::add_pointer_t`
- `template<typename _Tp>`  
`using std::add_rvalue_reference_t`
- `template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>`  
`using std::aligned_storage_t`
- `template<size_t _Len, typename... _Types>`  
`using std::aligned_union_t`
- `template<typename... _Tp>`  
`using std::common_type_t`
- `template<bool _Cond, typename _Iftrue, typename _Iffalse>`  
`using std::conditional_t`

- `template<typename _Tp>`  
using [std::decay\\_t](#)
- `template<bool _Cond, typename _Tp = void>`  
using [std::enable\\_if\\_t](#)
- using [std::false\\_type](#)
- `template<typename _Tp>`  
using [std::make\\_signed\\_t](#)
- `template<typename _Tp>`  
using [std::make\\_unsigned\\_t](#)
- `template<typename _Tp>`  
using [std::remove\\_all\\_extents\\_t](#)
- `template<typename _Tp>`  
using [std::remove\\_extent\\_t](#)
- `template<typename _Tp>`  
using [std::remove\\_pointer\\_t](#)
- `template<typename _Tp>`  
using [std::remove\\_reference\\_t](#)
- `template<typename _Tp>`  
using [std::result\\_of\\_t](#)
- using [std::true\\_type](#)
- `template<typename _Tp>`  
using [std::underlying\\_type\\_t](#)

## Functions

- `template<typename _Tp>`  
auto [std::declval](#)() noexcept -> decltype(\_\_declval<\_Tp>())
- `template<typename _Tp>`  
constexpr Require<\_\_not\_<\_\_is\_tuple\_like<\_Tp>>, [is\\_move\\_constructible](#)<\_Tp>, [is\\_move\\_assignable](#)<\_Tp>> [std::swap](#)(\_Tp&, \_Tp&) noexcept(\_\_and\_<[is\\_nothrow\\_move\\_constructible](#)<\_Tp>, [is\\_nothrow\\_move\\_assignable](#)<\_Tp>>::value)
- `template<typename _Tp, size_t _Nm>`  
constexpr \_\_enable\_if\_t<\_\_is\_swappable<\_Tp>::value> [std::swap](#)(\_Tp(&\_\_a)[\_Nm], \_Tp(&\_\_b)[\_Nm]) noexcept(\_\_is\_nothrow\_swappable<\_Tp>::value)

## Variables

- `template<typename _Tp, typename... _Args>`  
constexpr bool [std::\\_\\_is\\_nothrow\\_new\\_constructible](#)
- `template<size_t _Len, typename... _Types>`  
const size\_t [std::aligned\\_union](#)<\_Len, \_Types...>::alignment\_value
- `template<typename... _Tp>`  
using [std::common\\_reference\\_t](#)
- `template<typename _S1, typename _S2, typename _M1, typename _M2>`  
constexpr bool [std::is\\_corresponding\\_member](#)(\_M1 \_S1::\*\_\_m1, \_M2 \_S2::\*\_\_m2) noexcept
- `template<typename _Tp, typename _Up>`  
constexpr bool [std::is\\_layout\\_compatible\\_v](#)
- `template<typename _Base, typename _Derived>`  
constexpr bool [std::is\\_pointer\\_interconvertible\\_base\\_of\\_v](#)
- `template<typename _Tp, typename _Mem>`  
constexpr bool [std::is\\_pointer\\_interconvertible\\_with\\_class](#)(\_Mem \_Tp::\*\_\_mp) noexcept

### 6.688.1 Detailed Description

This is a Standard C++ Library header.

## 6.689 typeindex File Reference

### Classes

- struct [std::hash< type\\_index >](#)
- struct [std::type\\_index](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_TYPEINDEX`

### 6.689.1 Detailed Description

This is a Standard C++ Library header.

## 6.690 debug/unordered\_map File Reference

### Classes

- class [std::\\_\\_debug::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)
- class [std::\\_\\_debug::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Macros

- `#define _GLIBCXX_DEBUG_UNORDERED_MAP`

### Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
bool std::__debug::operator== (const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
bool std::__debug::operator== (const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
void std::__debug::swap (unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
void std::__debug::swap (unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `unordered_map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, hash<__iter_key_t<_InputIterator>>, equal_to<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`_InputIterator`, `_InputIterator`, `typename unordered_map<int, int>::size_type`, `_Allocator`) -> `unordered_map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, hash<__iter_key_t<_InputIterator>>, equal_to<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`_InputIterator`, `_InputIterator`, `typename unordered_map<int, int>::size_type`, `_Hash`, `_Allocator`) -> `unordered_map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Hash, equal_to<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`_InputIterator`, `_InputIterator`, `typename unordered_map<int, int>::size_type`, `_Hash`, `_Pred`, `_Allocator`) -> `unordered_map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Hash, _Pred, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`initializer_list<pair<_Key, _Tp>>`, `_Allocator`) -> `unordered_map<_Key, _Tp, hash<_Key>, equal_to<_Key>, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`initializer_list<pair<_Key, _Tp>>`, `typename unordered_map<int, int>::size_type`, `_Allocator`) -> `unordered_map<_Key, _Tp, hash<_Key>, equal_to<_Key>, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`initializer_list<pair<_Key, _Tp>>`, `typename unordered_map<int, int>::size_type`, `_Hash`, `_Allocator`) -> `unordered_map<_Key, _Tp, _Hash, equal_to<_Key>, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`initializer_list<pair<_Key, _Tp>>`, `typename unordered_map<int, int>::size_type`, `_Hash`, `_Pred`, `_Allocator`) -> `unordered_map<_Key, _Tp, _Hash, _Pred, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `unordered_multimap<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, hash<__iter_key_t<_InputIterator>>, equal_to<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap<int, int>::size_type`, `_Allocator`) -> `unordered_multimap<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, hash<__iter_key_t<_InputIterator>>, equal_to<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap<int, int>::size_type`, `_Hash`, `_Allocator`) -> `unordered_multimap<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Hash, equal_to<__iter_key_t<_InputIterator>>, _Allocator>`

- `template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multimap` (`_InputIterator`, `_InputIterator`, `unordered_multimap< int, int >::size_type`, `{}`, `_Hash`, `_Pred`, `_Allocator`) -> `unordered_multimap< __iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Hash, _Pred, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multimap` (`initializer_list< pair< _Key, _Tp >>`, `_Allocator`) -> `unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multimap` (`initializer_list< pair< _Key, _Tp >>`, `unordered_multimap< int, int >::size_type`, `_Allocator`) -> `unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multimap` (`initializer_list< pair< _Key, _Tp >>`, `unordered_multimap< int, int >::size_type`, `_Hash`, `_Allocator`) -> `unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multimap` (`initializer_list< pair< _Key, _Tp >>`, `unordered_multimap< int, int >::size_type`, `{}`, `_Hash`, `_Pred`, `_Allocator`) -> `unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator>`

### 6.690.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.691 experimental/unordered\_map File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_UNORDERED_MAP`

### Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>>`  
using `std::experimental::fundamentals_v2::pmr::unordered_map`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>>`  
using `std::experimental::fundamentals_v2::pmr::unordered_multimap`

### Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
void `std::experimental::erase_if` (`unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont`, `_Predicate __pred`)
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
void `std::experimental::erase_if` (`unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont`, `_Predicate __pred`)

### 6.691.1 Detailed Description

This is a TS C++ Library header.

## 6.692 unordered\_map File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_containers_ranges`
- `#define __glibcxx_want_erase_if`
- `#define __glibcxx_want_generic_unordered_lookup`
- `#define __glibcxx_want_node_extract`
- `#define __glibcxx_want_nonmember_container_access`
- `#define __glibcxx_want_tuple_like`
- `#define __glibcxx_want_unordered_map_try_emplace`
- `#define _GLIBCXX_UNORDERED_MAP`

### Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>  
using std::pmr::unordered\_map`
- `template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>  
using std::pmr::unordered\_multimap`

### Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>  
unordered\_map< _Key, _Tp, _Hash, _CPred, _Alloc >::size_type std::erase\_if (unordered\_map< _Key, _Tp,  
_Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>  
unordered\_multimap< _Key, _Tp, _Hash, _CPred, _Alloc >::size_type std::erase\_if (unordered\_multimap< ↵  
_Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`

### 6.692.1 Detailed Description

This is a Standard C++ Library header.

## 6.693 debug/unordered\_set File Reference

### Classes

- class [std::\\_\\_debug::unordered\\_multiset](#)< \_Value, \_Hash, \_Pred, \_Alloc >
- class [std::\\_\\_debug::unordered\\_set](#)< \_Value, \_Hash, \_Pred, \_Alloc >

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)



## Macros

- `#define _GLIBCXX_DEBUG_UNORDERED_SET`

## Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__debug::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__debug::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multiset ( _InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multiset ( _InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multiset ( _InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multiset ( _InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >`
- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >`
- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >`
- `template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >`
- `template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>`

```

std::__debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=↵
↵_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >
• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↵
↵Allocator<_Allocator>>
std::__debug::unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename
↵iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type
↵, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↵
↵Allocator<_Allocator>>
std::__debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) ->
↵unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< ↵
↵_InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator
↵ >
• template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↵
↵_RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
std::__debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, ↵
↵Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< type-
↵name iterator_traits< _InputIterator >::value_type >, _Allocator >
• template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred
↵= equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_Input↵
↵Iterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename
↵= _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
std::__debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=↵
↵Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator
↵ >::value_type, _Hash, _Pred, _Allocator >
• template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
std::__debug::unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >,
↵equal_to< _Tp >, _Allocator >
• template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
std::__debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) ->
↵unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
• template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = ↵
↵_RequireAllocator<_Allocator>>
std::__debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) ->
↵unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >
• template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>,
↵typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<↵
↵Allocator>>
std::__debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), ↵
↵_Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >

```

### 6.693.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.694 experimental/unordered\_set File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define GLIBCXX_EXPERIMENTAL_UNORDERED_SET`

## Typedefs

- `template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
using **`std::experimental::fundamentals_v2::pmr::unordered_multiset`**
- `template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
using **`std::experimental::fundamentals_v2::pmr::unordered_set`**

## Functions

- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
void **`std::experimental::erase_if`** (`unordered_multiset`< \_Key, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
void **`std::experimental::erase_if`** (`unordered_set`< \_Key, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)

### 6.694.1 Detailed Description

This is a TS C++ Library header.

## 6.695 unordered\_set File Reference

### Namespaces

- namespace `std`

### Macros

- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_containers_ranges`
- `#define __glibcxx_want_erase_if`
- `#define __glibcxx_want_generic_unordered_lookup`
- `#define __glibcxx_want_node_extract`
- `#define __glibcxx_want_nonmember_container_access`
- `#define _GLIBCXX_UNORDERED_SET`

## Typedefs

- `template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>`  
using **`std::pmr::unordered_multiset`**
- `template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>`  
using **`std::pmr::unordered_set`**

## Functions

- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`unordered_multiset`< \_Key, \_Hash, \_CPred, \_Alloc >::size\_type **`std::erase_if`** (`unordered_multiset`< \_Key, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`unordered_set`< \_Key, \_Hash, \_CPred, \_Alloc >::size\_type **`std::erase_if`** (`unordered_set`< \_Key, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)

### 6.695.1 Detailed Description

This is a Standard C++ Library header.

## 6.696 experimental/utility File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_UTILITY`

### Typedefs

- using `std::experimental::erased_type`

#### 6.696.1 Detailed Description

This is a TS C++ Library header.

## 6.697 utility File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_addressof_constexpr`
- `#define __glibcxx_want_as_const`
- `#define __glibcxx_want_constexpr_algorithms`
- `#define __glibcxx_want_constexpr_utility`
- `#define __glibcxx_want_constrained_equality`
- `#define __glibcxx_want_exchange_function`
- `#define __glibcxx_want_forward_like`
- `#define __glibcxx_want_integer_comparison_functions`
- `#define __glibcxx_want_integer_sequence`
- `#define __glibcxx_want_ranges_zip`
- `#define __glibcxx_want_to_underlying`
- `#define __glibcxx_want_tuple_element_t`
- `#define __glibcxx_want_tuple_like`
- `#define __glibcxx_want_tuples_by_type`
- `#define __glibcxx_want_unreachable`
- `#define _GLIBCXX_UTILITY`

#### 6.697.1 Detailed Description

This is a Standard C++ Library header.

## 6.698 valarray File Reference

### Classes

- class [std::valarray<\\_Tp>](#)

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

## Macros

- `#define _GLIBCXX_VALARRAY`

## Functions

- `template<class _Tp>`  
`const _Tp * std::begin (const valarray< _Tp > &__va) noexcept`
- `template<class _Tp>`  
`_Tp * std::begin (valarray< _Tp > &__va) noexcept`
- `template<class _Tp>`  
`const _Tp * std::end (const valarray< _Tp > &__va) noexcept`
- `template<class _Tp>`  
`_Tp * std::end (valarray< _Tp > &__va) noexcept`
- `template<typename _Tp, size_t _Nm>`  
`std::valarray (const _Tp(&)[_Nm], size_t) -> valarray< _Tp >`

### 6.698.1 Detailed Description

This is a Standard C++ Library header.

## 6.699 variant File Reference

### Macros

- `#define __glibcxx_want_constrained_equality`
- `#define __glibcxx_want_freestanding_variant`
- `#define __glibcxx_want_variant`
- `#define _GLIBCXX_VARIANT`

### 6.699.1 Detailed Description

This is the `<variant>` C++ Library header.

## 6.700 debug/vector File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_vector](#)< [\\_SafeSequence](#), [\\_BaseSequence](#) >
- struct [std::hash](#)< [\\_\\_debug::vector](#)< [bool](#), [\\_Alloc](#) > >
- class [std::\\_\\_debug::vector](#)< [\\_Tp](#), [\\_Allocator](#) >

## Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)
- namespace [std::\\_\\_detail](#)

## Macros

- `#define _GLIBCXX_DEBUG_VECTOR`

## Functions

- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp > std::__debug::operator<=> (const vector< _Tp, _Alloc > &__x,`  
`const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr bool std::__debug::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc >`  
`&__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr void std::__debug::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`  
`noexcept(/*conditional */)`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator =`  
`allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::vector (size_t, _Tp, _Allocator=_Allocator()) -> vector< _Tp, _Allocator >`

### 6.700.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.701 experimental/vector File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_erase_if`
- `#define _GLIBCXX_EXPERIMENTAL_VECTOR`

### Typedefs

- `template<typename _Tp>`  
`using std::experimental::fundamentals_v2::pmr::vector`

### Functions

- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void std::experimental::erase (vector< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (vector< _Tp, _Alloc > &__cont, _Predicate __pred)`

### 6.701.1 Detailed Description

This is a TS C++ Library header.

## 6.702 vector File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define __glibcxx_want_algorithm_default_value_type`
- `#define __glibcxx_want_allocator_traits_is_always_equal`
- `#define __glibcxx_want_constexpr_vector`
- `#define __glibcxx_want_containers_ranges`
- `#define __glibcxx_want_erase_if`
- `#define __glibcxx_want_incomplete_container_elements`
- `#define __glibcxx_want_nonmember_container_access`
- `#define _GLIBCXX_VECTOR`

### Typedefs

- `template<typename _Tp>`  
`using std::pmr::vector`

#### 6.702.1 Detailed Description

This is a Standard C++ Library header.

## 6.703 atomic\_word.h File Reference

### Macros

- `#define _GLIBCXX_READ_MEM_BARRIER`
- `#define _GLIBCXX_WRITE_MEM_BARRIER`

### Typedefs

- `typedef int _Atomic_word`

#### 6.703.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.704 basic\_file.h File Reference

### Namespaces

- namespace [std](#)

#### 6.704.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.705 c++allocator.h File Reference

### Namespaces

- namespace [std](#)

## Typedefs

- `template<typename _Tp>`  
using `std::__allocator_base`

### 6.705.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.706 c++config.h File Reference

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Macros

- `#define __GLIBCXX__`
- `#define __glibcxx_assert(cond)`
- `#define __N(msgid)`
- `#define _GLIBCXX11_DEPRECATED`
- `#define _GLIBCXX11_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX11_USE_C99_MATH`
- `#define _GLIBCXX11_USE_C99_STDIO`
- `#define _GLIBCXX11_USE_C99_STDLIB`
- `#define _GLIBCXX11_USE_C99_WCHAR`
- `#define _GLIBCXX14_DEPRECATED`
- `#define _GLIBCXX14_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX17_DEPRECATED`
- `#define _GLIBCXX17_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX20_DEPRECATED`
- `#define _GLIBCXX20_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX23_DEPRECATED`
- `#define _GLIBCXX23_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX26_CONSTEXPR`
- `#define _GLIBCXX26_DEPRECATED`
- `#define _GLIBCXX26_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX98_USE_C99_COMPLEX`
- `#define _GLIBCXX98_USE_C99_MATH`
- `#define _GLIBCXX98_USE_C99_STDIO`
- `#define _GLIBCXX98_USE_C99_STDLIB`
- `#define _GLIBCXX98_USE_C99_WCHAR`
- `#define _GLIBCXX_ABI_TAG_CXX11`
- `#define _GLIBCXX_ASSERT_FAIL(_Condition)`
- `#define _GLIBCXX_ASSERTIONS`
- `#define _GLIBCXX_ATOMIC_WORD_BUILTINS`
- `#define _GLIBCXX_AUTO_CAST(X)`
- `#define _GLIBCXX_BEGIN_EXTERN_C`
- `#define _GLIBCXX_BEGIN_INLINE_ABI_NAMESPACE(X)`
- `#define _GLIBCXX_BEGIN_NAMESPACE_ALGO`
- `#define _GLIBCXX_BEGIN_NAMESPACE_CONTAINER`



- #define \_GLIBCXX\_BEGIN\_NAMESPACE\_CXX11
- #define \_GLIBCXX\_BEGIN\_NAMESPACE\_LDBL
- #define \_GLIBCXX\_BEGIN\_NAMESPACE\_LDBL\_OR\_CXX11
- #define \_GLIBCXX\_BEGIN\_NAMESPACE\_VERSION
- #define \_GLIBCXX\_CAN\_ALIGNAS\_DESTRUCTIVE\_SIZE
- #define \_GLIBCXX\_DARWIN\_USE\_64\_BIT\_INODE
- #define \_GLIBCXX\_DEFAULT\_ABI\_TAG
- #define \_GLIBCXX\_DEPRECATED
- #define \_GLIBCXX\_DEPRECATED\_SUGGEST(ALT)
- #define \_GLIBCXX\_DOXYGEN\_ONLY(X)
- #define \_GLIBCXX\_END\_EXTERN\_C
- #define \_GLIBCXX\_END\_INLINE\_ABI\_NAMESPACE(X)
- #define \_GLIBCXX\_END\_NAMESPACE\_ALGO
- #define \_GLIBCXX\_END\_NAMESPACE\_CONTAINER
- #define \_GLIBCXX\_END\_NAMESPACE\_CXX11
- #define \_GLIBCXX\_END\_NAMESPACE\_LDBL
- #define \_GLIBCXX\_END\_NAMESPACE\_LDBL\_OR\_CXX11
- #define \_GLIBCXX\_END\_NAMESPACE\_VERSION
- #define \_GLIBCXX\_EXTERN\_TEMPLATE
- #define \_GLIBCXX\_EXTERN\_TEMPLATE
- #define \_GLIBCXX\_FAST\_MATH
- #define \_GLIBCXX\_FULLY\_DYNAMIC\_STRING
- #define \_GLIBCXX\_HAS\_BUILTIN(B)
- #define \_GLIBCXX\_HAVE\_\_\_CXA\_THREAD\_ATEXIT\_IMPL
- #define \_GLIBCXX\_HAVE\_ACOSF
- #define \_GLIBCXX\_HAVE\_ACOSL
- #define \_GLIBCXX\_HAVE\_ALIGNED\_ALLOC
- #define \_GLIBCXX\_HAVE\_ARC4RANDOM
- #define \_GLIBCXX\_HAVE\_ARPA\_INET\_H
- #define \_GLIBCXX\_HAVE\_AS\_SYMVER\_DIRECTIVE
- #define \_GLIBCXX\_HAVE\_ASINF
- #define \_GLIBCXX\_HAVE\_ASINL
- #define \_GLIBCXX\_HAVE\_AT\_QUICK\_EXIT
- #define \_GLIBCXX\_HAVE\_ATAN2F
- #define \_GLIBCXX\_HAVE\_ATAN2L
- #define \_GLIBCXX\_HAVE\_ATANF
- #define \_GLIBCXX\_HAVE\_ATANL
- #define \_GLIBCXX\_HAVE\_ATOMIC\_LOCK\_POLICY
- #define \_GLIBCXX\_HAVE\_ATTRIBUTE\_VISIBILITY
- #define \_GLIBCXX\_HAVE\_BUILTIN\_HAS\_UNIQ\_OBJ\_REP
- #define \_GLIBCXX\_HAVE\_BUILTIN\_IS\_AGGREGATE
- #define \_GLIBCXX\_HAVE\_BUILTIN\_LAUNDER
- #define \_GLIBCXX\_HAVE\_C99\_FLT\_EVAL\_TYPES
- #define \_GLIBCXX\_HAVE\_CEILF
- #define \_GLIBCXX\_HAVE\_CEILL
- #define \_GLIBCXX\_HAVE\_COMPLEX\_H
- #define \_GLIBCXX\_HAVE\_COSF
- #define \_GLIBCXX\_HAVE\_COSHF
- #define \_GLIBCXX\_HAVE\_COSHL
- #define \_GLIBCXX\_HAVE\_COSL
- #define \_GLIBCXX\_HAVE\_DECL\_STRNLEN

- `#define _GLIBCXX_HAVE_DIRENT_H`
- `#define _GLIBCXX_HAVE_DIRFD`
- `#define _GLIBCXX_HAVE_DLFCN_H`
- `#define _GLIBCXX_HAVE_ENDIAN_H`
- `#define _GLIBCXX_HAVE_EXCEPTION_PTR_SINCE_GCC46`
- `#define _GLIBCXX_HAVE_EXECINFO_H`
- `#define _GLIBCXX_HAVE_EXPF`
- `#define _GLIBCXX_HAVE_EXPL`
- `#define _GLIBCXX_HAVE_FABSF`
- `#define _GLIBCXX_HAVE_FABSL`
- `#define _GLIBCXX_HAVE_FCNTL_H`
- `#define _GLIBCXX_HAVE_FDOPENDIR`
- `#define _GLIBCXX_HAVE_FENV_H`
- `#define _GLIBCXX_HAVE_FINITE`
- `#define _GLIBCXX_HAVE_FINITEF`
- `#define _GLIBCXX_HAVE_FINITEL`
- `#define _GLIBCXX_HAVE_FLOAT_H`
- `#define _GLIBCXX_HAVE_FLOORF`
- `#define _GLIBCXX_HAVE_FLOORL`
- `#define _GLIBCXX_HAVE_FMODF`
- `#define _GLIBCXX_HAVE_FMODL`
- `#define _GLIBCXX_HAVE_FREXPF`
- `#define _GLIBCXX_HAVE_FREXPL`
- `#define _GLIBCXX_HAVE_GETENTROPY`
- `#define _GLIBCXX_HAVE_GETIPINFO`
- `#define _GLIBCXX_HAVE_GETS`
- `#define _GLIBCXX_HAVE_HYPOT`
- `#define _GLIBCXX_HAVE_HYPOTF`
- `#define _GLIBCXX_HAVE_HYPOTL`
- `#define _GLIBCXX_HAVE_ICONV`
- `#define _GLIBCXX_HAVE_INTTYPES_H`
- `#define _GLIBCXX_HAVE_IS_CONSTANT_EVALUATED`
- `#define _GLIBCXX_HAVE_ISINF`
- `#define _GLIBCXX_HAVE_ISINFF`
- `#define _GLIBCXX_HAVE_ISINFL`
- `#define _GLIBCXX_HAVE_ISNAN`
- `#define _GLIBCXX_HAVE_ISNANF`
- `#define _GLIBCXX_HAVE_ISNANL`
- `#define _GLIBCXX_HAVE_ISWBLANK`
- `#define _GLIBCXX_HAVE_LC_MESSAGES`
- `#define _GLIBCXX_HAVE_LDEXPF`
- `#define _GLIBCXX_HAVE_LDEXPL`
- `#define _GLIBCXX_HAVE_LIBINTL_H`
- `#define _GLIBCXX_HAVE_LIMIT_AS`
- `#define _GLIBCXX_HAVE_LIMIT_DATA`
- `#define _GLIBCXX_HAVE_LIMIT_FSIZE`
- `#define _GLIBCXX_HAVE_LIMIT_RSS`
- `#define _GLIBCXX_HAVE_LIMIT_VMEM`
- `#define _GLIBCXX_HAVE_LINK`
- `#define _GLIBCXX_HAVE_LINK_H`
- `#define _GLIBCXX_HAVE_LINUX_FUTEX`

- #define \_GLIBCXX\_HAVE\_LINUX\_RANDOM\_H
- #define \_GLIBCXX\_HAVE\_LINUX\_TYPES\_H
- #define \_GLIBCXX\_HAVE\_LOCALE\_H
- #define \_GLIBCXX\_HAVE\_LOG10F
- #define \_GLIBCXX\_HAVE\_LOG10L
- #define \_GLIBCXX\_HAVE\_LOGF
- #define \_GLIBCXX\_HAVE\_LOGL
- #define \_GLIBCXX\_HAVE\_LSEEK
- #define \_GLIBCXX\_HAVE\_MBSTATE\_T
- #define \_GLIBCXX\_HAVE\_MEMALIGN
- #define \_GLIBCXX\_HAVE\_MEMORY\_H
- #define \_GLIBCXX\_HAVE\_MODFF
- #define \_GLIBCXX\_HAVE\_MODFL
- #define \_GLIBCXX\_HAVE\_NETDB\_H
- #define \_GLIBCXX\_HAVE\_NETINET\_IN\_H
- #define \_GLIBCXX\_HAVE\_NETINET\_TCP\_H
- #define \_GLIBCXX\_HAVE\_O\_NONBLOCK
- #define \_GLIBCXX\_HAVE\_OPENAT
- #define \_GLIBCXX\_HAVE\_POLL
- #define \_GLIBCXX\_HAVE\_POLL\_H
- #define \_GLIBCXX\_HAVE\_POSIX\_MEMALIGN
- #define \_GLIBCXX\_HAVE\_POWF
- #define \_GLIBCXX\_HAVE\_POWL
- #define \_GLIBCXX\_HAVE\_QUICK\_EXIT
- #define \_GLIBCXX\_HAVE\_READLINK
- #define \_GLIBCXX\_HAVE\_S\_ISREG
- #define \_GLIBCXX\_HAVE\_SECURE\_GETENV
- #define \_GLIBCXX\_HAVE\_SETENV
- #define \_GLIBCXX\_HAVE\_SINCOS
- #define \_GLIBCXX\_HAVE\_SINCOSF
- #define \_GLIBCXX\_HAVE\_SINCOSL
- #define \_GLIBCXX\_HAVE\_SINF
- #define \_GLIBCXX\_HAVE\_SINHF
- #define \_GLIBCXX\_HAVE\_SINHL
- #define \_GLIBCXX\_HAVE\_SINL
- #define \_GLIBCXX\_HAVE\_SOCKETATMARK
- #define \_GLIBCXX\_HAVE\_SQRTF
- #define \_GLIBCXX\_HAVE\_SQRTL
- #define \_GLIBCXX\_HAVE\_STACKTRACE
- #define \_GLIBCXX\_HAVE\_STDALIGN\_H
- #define \_GLIBCXX\_HAVE\_STDBOOL\_H
- #define \_GLIBCXX\_HAVE\_STDINT\_H
- #define \_GLIBCXX\_HAVE\_STDLIB\_H
- #define \_GLIBCXX\_HAVE\_STRERROR\_L
- #define \_GLIBCXX\_HAVE\_STRERROR\_R
- #define \_GLIBCXX\_HAVE\_STRING\_H
- #define \_GLIBCXX\_HAVE\_STRINGS\_H
- #define \_GLIBCXX\_HAVE\_STRTOF
- #define \_GLIBCXX\_HAVE\_STRTOLD
- #define \_GLIBCXX\_HAVE\_STRUCT\_DIRENT\_D\_TYPE
- #define \_GLIBCXX\_HAVE\_STRXFRM\_L

- `#define _GLIBCXX_HAVE_SYMLINK`
- `#define _GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT`
- `#define _GLIBCXX_HAVE_SYS_IOCTL_H`
- `#define _GLIBCXX_HAVE_SYS_IPC_H`
- `#define _GLIBCXX_HAVE_SYS_MMAN_H`
- `#define _GLIBCXX_HAVE_SYS_PARAM_H`
- `#define _GLIBCXX_HAVE_SYS_PTRACE_H`
- `#define _GLIBCXX_HAVE_SYS_RESOURCE_H`
- `#define _GLIBCXX_HAVE_SYS_SEM_H`
- `#define _GLIBCXX_HAVE_SYS_SOCKET_H`
- `#define _GLIBCXX_HAVE_SYS_STAT_H`
- `#define _GLIBCXX_HAVE_SYS_STATVFS_H`
- `#define _GLIBCXX_HAVE_SYS_SYSINFO_H`
- `#define _GLIBCXX_HAVE_SYS_TIME_H`
- `#define _GLIBCXX_HAVE_SYS_TYPES_H`
- `#define _GLIBCXX_HAVE_SYS_UIO_H`
- `#define _GLIBCXX_HAVE_TANF`
- `#define _GLIBCXX_HAVE_TANHF`
- `#define _GLIBCXX_HAVE_TANHL`
- `#define _GLIBCXX_HAVE_TANL`
- `#define _GLIBCXX_HAVE_TGMATH_H`
- `#define _GLIBCXX_HAVE_TIMESPEC_GET`
- `#define _GLIBCXX_HAVE_TLS`
- `#define _GLIBCXX_HAVE_TRUNCATE`
- `#define _GLIBCXX_HAVE_UCHAR_H`
- `#define _GLIBCXX_HAVE_UNISTD_H`
- `#define _GLIBCXX_HAVE_UNLINKAT`
- `#define _GLIBCXX_HAVE_USELOCALE`
- `#define _GLIBCXX_HAVE_UTIME_H`
- `#define _GLIBCXX_HAVE_VFWSCANF`
- `#define _GLIBCXX_HAVE_VSWSCANF`
- `#define _GLIBCXX_HAVE_VWSCANF`
- `#define _GLIBCXX_HAVE_WCHAR_H`
- `#define _GLIBCXX_HAVE_WCSTOF`
- `#define _GLIBCXX_HAVE_WCTYPE_H`
- `#define _GLIBCXX_HAVE_WRITEV`
- `#define _GLIBCXX_HOSTED`
- `#define _GLIBCXX_ICONV_CONST`
- `#define _GLIBCXX_INLINE_VERSION`
- `#define _GLIBCXX_LT_OBJDIR`
- `#define _GLIBCXX_MANGLE_SIZE_T`
- `#define _GLIBCXX_NAMESPACE_CXX11`
- `#define _GLIBCXX_NAMESPACE_LDBL`
- `#define _GLIBCXX_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_NODISCARD`
- `#define _GLIBCXX_NOEXCEPT_PARM`
- `#define _GLIBCXX_NOEXCEPT_QUAL`
- `#define _GLIBCXX_PACKAGE _GLIBCXX_VERSION`
- `#define _GLIBCXX_PACKAGE_BUGREPORT`
- `#define _GLIBCXX_PACKAGE_NAME`
- `#define _GLIBCXX_PACKAGE_STRING`

- `#define _GLIBCXX_PACKAGE_TARNAME`
- `#define _GLIBCXX_PACKAGE_URL`
- `#define _GLIBCXX_PSEUDO_VISIBILITY(V)`
- `#define _GLIBCXX_RELEASE`
- `#define _GLIBCXX_RES_LIMITS`
- `#define _GLIBCXX_STATIC_TZDATA`
- `#define _GLIBCXX_STD_A`
- `#define _GLIBCXX_STD_C`
- `#define _GLIBCXX_STDC_HEADERS`
- `#define _GLIBCXX_STDIO_EOF`
- `#define _GLIBCXX_STDIO_SEEK_CUR`
- `#define _GLIBCXX_STDIO_SEEK_END`
- `#define _GLIBCXX_SYMVER`
- `#define _GLIBCXX_SYMVER_GNU`
- `#define _GLIBCXX_SYNCHRONIZATION_HAPPENS_AFTER(A)`
- `#define _GLIBCXX_SYNCHRONIZATION_HAPPENS_BEFORE(A)`
- `#define _GLIBCXX_THROW_OR_ABORT(_EXC)`
- `#define _GLIBCXX_TXN_SAFE`
- `#define _GLIBCXX_TXN_SAFE_DYN`
- `#define _GLIBCXX_USE_ALLOCATOR_NEW`
- `#define _GLIBCXX_USE_BUILTIN_TRAIT(BT)`
- `#define _GLIBCXX_USE_C11_UCHAR_CXX11`
- `#define _GLIBCXX_USE_C99`
- `#define _GLIBCXX_USE_C99_COMPLEX`
- `#define _GLIBCXX_USE_C99_COMPLEX_TR1`
- `#define _GLIBCXX_USE_C99_CTYPE`
- `#define _GLIBCXX_USE_C99_CTYPE_TR1`
- `#define _GLIBCXX_USE_C99_FENV`
- `#define _GLIBCXX_USE_C99_FENV_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES`
- `#define _GLIBCXX_USE_C99_INTTYPES_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T`
- `#define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1`
- `#define _GLIBCXX_USE_C99_MATH`
- `#define _GLIBCXX_USE_C99_MATH_FUNCS`
- `#define _GLIBCXX_USE_C99_MATH_TR1`
- `#define _GLIBCXX_USE_C99_STDINT`
- `#define _GLIBCXX_USE_C99_STDINT_TR1`
- `#define _GLIBCXX_USE_C99_STDIO`
- `#define _GLIBCXX_USE_C99_STDLIB`
- `#define _GLIBCXX_USE_C99_WCHAR`
- `#define _GLIBCXX_USE_CHDIR`
- `#define _GLIBCXX_USE_CHMOD`
- `#define _GLIBCXX_USE_CLOCK_MONOTONIC`
- `#define _GLIBCXX_USE_CLOCK_REALTIME`
- `#define _GLIBCXX_USE_CXX11_ABI`
- `#define _GLIBCXX_USE_DECIMAL_FLOAT`
- `#define _GLIBCXX_USE_DEV_RANDOM`
- `#define _GLIBCXX_USE_DUAL_ABI`
- `#define _GLIBCXX_USE_FCHMOD`
- `#define _GLIBCXX_USE_FCHMODAT`

- #define `_GLIBCXX_USE_FSEEKO_FTELLO`
- #define `_GLIBCXX_USE_GET_NPROCS`
- #define `_GLIBCXX_USE_GETCWD`
- #define `_GLIBCXX_USE_GETTIMEOFDAY`
- #define `_GLIBCXX_USE_INIT_PRIORITY_ATTRIBUTE`
- #define `_GLIBCXX_USE_LFS`
- #define `_GLIBCXX_USE_LONG_LONG`
- #define `_GLIBCXX_USE_LSTAT`
- #define `_GLIBCXX_USE_MKDIR`
- #define `_GLIBCXX_USE_NANOSLEEP`
- #define `_GLIBCXX_USE_NL_LANGINFO_L`
- #define `_GLIBCXX_USE_NLS`
- #define `_GLIBCXX_USE_PROC_SELF_STATUS`
- #define `_GLIBCXX_USE_PTHREAD_COND_CLOCKWAIT`
- #define `_GLIBCXX_USE_PTHREAD_MUTEX_CLOCKLOCK`
- #define `_GLIBCXX_USE_PTHREAD_RWLOCK_CLOCKLOCK`
- #define `_GLIBCXX_USE_RANDOM_TR1`
- #define `_GLIBCXX_USE_REALPATH`
- #define `_GLIBCXX_USE_SC_NPROCESSORS_ONLN`
- #define `_GLIBCXX_USE_SCHED_YIELD`
- #define `_GLIBCXX_USE_SENDFILE`
- #define `_GLIBCXX_USE_ST_MTIM`
- #define `_GLIBCXX_USE_STD_SPEC_FUNCS`
- #define `_GLIBCXX_USE_STRUCT_TM_TM_ZONE`
- #define `_GLIBCXX_USE_TMPNAM`
- #define `_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_CXX20`
- #define `_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_FCHAR8_T`
- #define `_GLIBCXX_USE_UTCIME`
- #define `_GLIBCXX_USE_UTIMENSAT`
- #define `_GLIBCXX_USE_WCHAR_T`
- #define `_GLIBCXX_USE_WEAK_REF`
- #define `_GLIBCXX_VERBOSE`
- #define `_GLIBCXX_VERBOSE_ASSERT`
- #define `_GLIBCXX_VISIBILITY(V)`
- #define `_GLIBCXX_WEAK_DEFINITION`
- #define `_GLIBCXX_X86_RDRAND`
- #define `_GLIBCXX_X86_RDSEED`
- #define `_GLIBCXX_ZONEINFO_DIR`
- #define `_GTHREAD_USE_MUTEX_TIMEDLOCK`

## Typedefs

- typedef decltype(nullptr) `std::nullptr_t`
- typedef \_\_PTRDIFF\_TYPE\_\_ `std::ptrdiff_t`
- typedef \_\_SIZE\_TYPE\_\_ `std::size_t`

## Functions

- void `std::__glibcxx_assert_fail` (const char \*\_\_file, int \_\_line, const char \*\_\_function, const char \*\_\_condition) noexcept
- constexpr bool `std::__is_constant_evaluated` () noexcept
- void `std::__terminate` () noexcept

### 6.706.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<version>`.

## 6.707 `c++io.h` File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef FILE `std::__c_file`
- typedef `__gthread_mutex_t` `std::__c_lock`

### 6.707.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.708 `c++locale.h` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_C_LOCALE_GNU`
- `#define _GLIBCXX_NUM_CATEGORIES`

### Typedefs

- typedef `__locale_t` `std::__c_locale`

### Functions

- int `std::__convert_from_v` (const `__c_locale` & `__cloc`, char \* `__out`, const int `__size`, const char \* `__fmt`,...)

### 6.708.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.709 `c++locale_internal.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- Catalogs & `std::get_catalogs` ()

### 6.709.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.710 `parallel/compatibility.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Tp>`  
`_Tp __gnu_parallel::__add_omp` (volatile `_Tp *``__ptr`, `_Tp` `__addend`)
- `template<typename _Tp>`  
`bool __gnu_parallel::__cas_omp` (volatile `_Tp *``__ptr`, `_Tp` `__comparand`, `_Tp` `__replacement`)
- `template<typename _Tp>`  
`bool __gnu_parallel::__compare_and_swap` (volatile `_Tp *``__ptr`, `_Tp` `__comparand`, `_Tp` `__replacement`)
- `template<typename _Tp>`  
`_Tp __gnu_parallel::__fetch_and_add` (volatile `_Tp *``__ptr`, `_Tp` `__addend`)
- `void __gnu_parallel::__yield` ()

### 6.710.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations.

This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

## 6.711 `x86_64-pc-linux-gnu/bits/compatibility.h` File Reference

### 6.711.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

## 6.712 `cpu_defines.h` File Reference

### 6.712.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 6.713 `ctype_base.h` File Reference

### Classes

- struct [std::ctype\\_base](#)

### Namespaces

- namespace [std](#)

### 6.713.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.



## 6.714 ctype\_inline.h File Reference

### Namespaces

- namespace [std](#)

### 6.714.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.715 cxxabi\_tweaks.h File Reference

### Macros

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`
- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

### Typedefs

- `typedef void __cxxabiv1::__cxa_cdtor_return_type`
- `typedef void __cxxabiv1::__cxa_vec_ctor_return_type`

### Variables

- `__extension__ typedef int __cxxabiv1::__guard`

### 6.715.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

## 6.716 error\_constants.h File Reference

### Namespaces

- namespace [std](#)

### Enumerations

- `enum class errc {`  
    `address_family_not_supported , address_in_use , address_not_available , already_connected ,`  
    `argument_list_too_long , argument_out_of_domain , bad_address , bad_file_descriptor ,`  
    `broken_pipe , connection_aborted , connection_already_in_progress , connection_refused ,`  
    `connection_reset , cross_device_link , destination_address_required , device_or_resource_busy ,`  
    `directory_not_empty , executable_format_error , file_exists , file_too_large ,`  
    `filename_too_long , function_not_supported , host_unreachable , illegal_byte_sequence ,`  
    `inappropriate_io_control_operation , interrupted , invalid_argument , invalid_seek ,`  
    `io_error , is_a_directory , message_size , network_down ,`  
    `network_reset , network_unreachable , no_buffer_space , no_child_process ,`  
    `no_lock_available , no_message , no_protocol_option , no_space_on_device ,`  
    `no_such_device_or_address , no_such_device , no_such_file_or_directory , no_such_process ,`  
    `...`  
};

`not_a_directory` , `not_a_socket` , `not_connected` , `not_enough_memory` ,  
`operation_in_progress` , `operation_not_permitted` , `operation_not_supported` , `operation_would_block` ,  
`permission_denied` , `protocol_not_supported` , `read_only_file_system` , `resource_deadlock_would_occur`  
,  
`resource_unavailable_try_again` , `result_out_of_range` , `timed_out` , `too_many_files_open_in_system` ,  
`too_many_files_open` , `too_many_links` , `too_many_symbolic_link_levels` , `wrong_protocol_type` }

#### 6.716.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<system_error>`.

### 6.717 extc++.h File Reference

#### 6.717.1 Detailed Description

This is an implementation file for a precompiled header.

### 6.718 messages\_members.h File Reference

#### Namespaces

- namespace [std](#)

#### 6.718.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 6.719 opt\_random.h File Reference

#### Namespaces

- namespace [std](#)

#### 6.719.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

### 6.720 os\_defines.h File Reference

#### Macros

- `#define __NO_CTYPE`
- `#define _GLIBCXX_MAY_HAVE__CXA_THREAD_ATEXIT_IMPL`
- `#define _GLIBCXX_NO_OBSOLETE_ISINF_ISNAN_DYNAMIC`

#### 6.720.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 6.721 `stdc++.h` File Reference

### 6.721.1 Detailed Description

This is an implementation file for a precompiled header.

## 6.722 `stdtr1c++.h` File Reference

### 6.722.1 Detailed Description

This is an implementation file for a precompiled header.

## 6.723 `time_members.h` File Reference

### Namespaces

- namespace [std](#)

### 6.723.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.724 `compare` File Reference

### Classes

- struct [std::compare\\_three\\_way\\_result<\\_Tp, \\_Up>](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Typedefs

- `template<typename _Tp, typename _Up>`  
`using std::\_\_detail::\_\_cmp3way\_res\_t`
- `template<typename... _Ts>`  
`using std::common\_comparison\_category\_t`
- `template<typename _Tp, typename _Up = _Tp>`  
`using std::compare\_three\_way\_result\_t`
- `using std::\_\_cmp\_cat::type`

### Enumerations

- `enum class \_Ord : type { equivalent , less , greater , unordered }`

### Functions

- `template<typename... _Ts>`  
`constexpr auto std::\_\_detail::\_\_common\_cmp\_cat ()`
- `template<typename _Ordering>`  
`constexpr _Ordering std::\_\_cmp\_cat::\_\_make (_Ord __o) noexcept`
- `template<typename _Ordering>`  
`constexpr _Ord std::\_\_cmp\_cat::\_\_ord (_Ordering __o) noexcept`

- constexpr bool **std::is\_eq** (partial\_ordering \_\_cmp) noexcept
- constexpr bool **std::is\_gt** (partial\_ordering \_\_cmp) noexcept
- constexpr bool **std::is\_gteq** (partial\_ordering \_\_cmp) noexcept
- constexpr bool **std::is\_lt** (partial\_ordering \_\_cmp) noexcept
- constexpr bool **std::is\_lteq** (partial\_ordering \_\_cmp) noexcept
- constexpr bool **std::is\_neq** (partial\_ordering \_\_cmp) noexcept

## Variables

- template<typename \_Tp>  
constexpr unsigned **std::\_\_detail::\_\_cmp\_cat\_id**
- template<> constexpr unsigned **std::\_\_detail::\_\_cmp\_cat\_id< partial\_ordering >**
- template<> constexpr unsigned **std::\_\_detail::\_\_cmp\_cat\_id< strong\_ordering >**
- template<> constexpr unsigned **std::\_\_detail::\_\_cmp\_cat\_id< weak\_ordering >**
- constexpr \_\_compare::\_\_Partial\_fallback **std::compare\_partial\_order\_fallback**
- constexpr \_\_compare::\_\_Strong\_fallback **std::compare\_strong\_order\_fallback**
- constexpr \_\_compare::\_\_Weak\_fallback **std::compare\_weak\_order\_fallback**
- constexpr \_\_compare::\_\_Partial\_order **std::partial\_order**
- constexpr \_\_compare::\_\_Strong\_order **std::strong\_order**
- constexpr \_\_compare::\_\_Weak\_order **std::weak\_order**

### 6.724.1 Detailed Description

This is a Standard C++ Library header.

## 6.725 cxxabi.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::recursive\\_init\\_error](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [abi](#)

### Typedefs

- typedef [\\_\\_cxa\\_cdtor\\_return\\_type](#)(\* [\\_\\_cxxabiv1::\\_\\_cxa\\_cdtor\\_type](#)) (void \*)

### Functions

- [\\_\\_cxa\\_dependent\\_exception](#) \* [\\_\\_cxxabiv1::\\_\\_cxa\\_allocate\\_dependent\\_exception](#) () noexcept
- int [\\_\\_cxxabiv1::\\_\\_cxa\\_atexit](#) (void\*)(void \*), void \*, void \*) noexcept
- void [\\_\\_cxxabiv1::\\_\\_cxa\\_bad\\_cast](#) ()
- void [\\_\\_cxxabiv1::\\_\\_cxa\\_bad\\_typeid](#) ()
- void \* [\\_\\_cxxabiv1::\\_\\_cxa\\_begin\\_catch](#) (void \*) noexcept
- [std::type\\_info](#) \* [\\_\\_cxxabiv1::\\_\\_cxa\\_current\\_exception\\_type](#) () noexcept
- void [\\_\\_cxxabiv1::\\_\\_cxa\\_deleted\\_virtual](#) (void)
- char \* [\\_\\_cxxabiv1::\\_\\_cxa\\_demangle](#) (const char \* \_\_mangled\_name, char \* \_\_output\_buffer, size\_t \* \_\_length, int \* \_\_status)
- void [\\_\\_cxxabiv1::\\_\\_cxa\\_end\\_catch](#) ()
- void [\\_\\_cxxabiv1::\\_\\_cxa\\_finalize](#) (void \*)

- void **\_\_cxxabiv1::\_\_cxa\_free\_dependent\_exception** (\_\_cxa\_dependent\_exception \*) noexcept
- void **\_\_cxxabiv1::\_\_cxa\_free\_exception** (void \*) noexcept
- void \* **\_\_cxxabiv1::\_\_cxa\_get\_exception\_ptr** (void \*) noexcept
- \_\_cxa\_eh\_globals \* **\_\_cxxabiv1::\_\_cxa\_get\_globals** () noexcept
- \_\_cxa\_eh\_globals \* **\_\_cxxabiv1::\_\_cxa\_get\_globals\_fast** () noexcept
- void **\_\_cxxabiv1::\_\_cxa\_guard\_abort** (\_\_guard \*) noexcept
- int **\_\_cxxabiv1::\_\_cxa\_guard\_acquire** (\_\_guard \*)
- void **\_\_cxxabiv1::\_\_cxa\_guard\_release** (\_\_guard \*) noexcept
- void **\_\_cxxabiv1::\_\_cxa\_pure\_virtual** (void)
- void **\_\_cxxabiv1::\_\_cxa\_rethrow** ()
- int **\_\_cxxabiv1::\_\_cxa\_thread\_atexit** (void\*)(void \*), void \*, void \*) noexcept
- void **\_\_cxxabiv1::\_\_cxa\_throw** (void \*, [std::type\\_info](#) \*, void\*)(void \*)
- void **\_\_cxxabiv1::\_\_cxa\_throw\_bad\_array\_new\_length** ()
- \_\_cxa\_vec\_ctor\_return\_type **\_\_cxxabiv1::\_\_cxa\_vec\_ctor** (void \* \_\_dest\_array, void \* \_\_src\_array, size\_t \_\_↵  
element\_count, size\_t \_\_element\_size, \_\_cxa\_ctor\_return\_type(\* \_\_constructor)(void \*, void \*), \_\_cxa\_ctor↵  
\_\_type \_\_destructor)
- void **\_\_cxxabiv1::\_\_cxa\_vec\_cleanup** (void \* \_\_array\_address, size\_t \_\_element\_count, size\_t \_\_s, \_\_cxa↵  
\_\_ctor\_type \_\_destructor) noexcept
- \_\_cxa\_vec\_ctor\_return\_type **\_\_cxxabiv1::\_\_cxa\_vec\_ctor** (void \* \_\_array\_address, size\_t \_\_element\_count,  
size\_t \_\_element\_size, \_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor)
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete** (void \* \_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, ↵  
\_\_cxa\_ctor\_type \_\_destructor)
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete2** (void \* \_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_destructor, void(\* \_\_dealloc)(void \*))
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete3** (void \* \_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_destructor, void(\* \_\_dealloc)(void \*, size\_t))
- void **\_\_cxxabiv1::\_\_cxa\_vec\_dtor** (void \* \_\_array\_address, size\_t \_\_element\_count, size\_t \_\_element\_size, ↵  
\_\_cxa\_ctor\_type \_\_destructor)
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor)
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new2** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor, void \*(\* \_\_alloc)(size\_t), void(\* \_\_dealloc)(void  
\*))
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new3** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size,  
\_\_cxa\_ctor\_type \_\_constructor, \_\_cxa\_ctor\_type \_\_destructor, void \*(\* \_\_alloc)(size\_t), void(\* \_\_dealloc)(void  
\*, size\_t))
- void \* **\_\_cxxabiv1::\_\_dynamic\_cast** (const void \* \_\_src\_ptr, const \_\_class\_type\_info \* \_\_src\_type, const \_\_↵  
class\_type\_info \* \_\_dst\_type, ptrdiff\_t \_\_src2dst)

### 6.725.1 Detailed Description

The header provides an interface to the C++ ABI.

### 6.725.2 Function Documentation

#### **\_\_cxa\_demangle()**

```
char * __cxxabiv1::__cxa_demangle (
    const char * __mangled_name,
    char * __output_buffer,
    size_t * __length,
    int * __status)
```

Demangling routine. ABI-mandated entry point in the C++ runtime library for demangling.

## Parameters

<code>__mangled_name</code>	A NUL-terminated character string containing the name to be demangled.
<code>__output_buffer</code>	A region of memory, allocated with <code>malloc</code> , of <code>*__length</code> bytes, into which the demangled name is stored. If <code>__output_buffer</code> is not long enough, it is expanded using <code>realloc</code> . <code>__output_buffer</code> may instead be null; in that case, the demangled name is placed in a region of memory allocated with <code>malloc</code> .
<code>__length</code>	If <code>__length</code> is non-null, the length of the buffer containing the demangled name is placed in <code>*__length</code> .
<code>__status</code>	If <code>__status</code> is non-null, <code>*__status</code> is set to one of the following values: 0: The demangling operation succeeded. -1: A memory allocation failure occurred. -2: <code>mangled_name</code> is not a valid name under the C++ ABI mangling rules. -3: One of the arguments is invalid.

## Returns

A pointer to the start of the NUL-terminated demangled name, or a null pointer if the demangling fails. The caller is responsible for deallocating this memory using `free`.

The demangling is performed using the C++ ABI mangling rules, with GNU extensions. For example, this function is used in `__gnu_cxx::__verbose_terminate_handler`.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext\\_demangling.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_demangling.html) for other examples of use.

## Note

The same demangling functionality is available via `libiberty` (`<libiberty/demangle.h>` and `libiberty.a`) in GCC 3.1 and later, but that requires explicit installation (`-enable-install-libiberty`) and uses a different API, although the ABI is unchanged.

## 6.726 exception File Reference

## Classes

- class [std::bad\\_exception](#)

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Typedefs

- typedef void(\* [std::terminate\\_handler](#)) ()
- typedef void(\* [std::unexpected\\_handler](#)) ()

## Functions

- void [\\_\\_gnu\\_cxx::\\_\\_verbose\\_terminate\\_handler](#) ()
- [terminate\\_handler](#) [std::get\\_terminate](#) () noexcept
- [unexpected\\_handler](#) [std::get\\_unexpected](#) () noexcept
- [terminate\\_handler](#) [std::set\\_terminate](#) ([terminate\\_handler](#)) noexcept
- [unexpected\\_handler](#) [std::set\\_unexpected](#) ([unexpected\\_handler](#)) noexcept
- void [std::terminate](#) () noexcept
- bool [std::uncaught\\_exception](#) () noexcept
- void [std::unexpected](#) ()

### 6.726.1 Detailed Description

This is a Standard C++ Library header.

## 6.727 `initializer_list` File Reference

### Classes

- class `std::initializer_list<_E>`

### Namespaces

- namespace `std`

### 6.727.1 Detailed Description

This is a Standard C++ Library header.

## 6.728 `new` File Reference

### Classes

- class `std::bad_alloc`
- struct `std::destroying_delete_t`

### Namespaces

- namespace `std`

### Macros

- `#define __glibcxx_want_constexpr_new`
- `#define __glibcxx_want_destroying_delete`
- `#define __glibcxx_want_hardware_interference_size`
- `#define __glibcxx_want_launder`
- `#define _GLIBCXX_PLACEMENT_CONSTEXPR`
- `#define _NEW`

### Typedefs

- typedef void(\* `std::new_handler`) ()

### Enumerations

- enum class `align_val_t` : `size_t`

### Functions

- `new_handler` `std::get_new_handler` () noexcept
- void **operator delete** (void \*) noexcept
- void **operator delete** (void \*, const std::nothrow\_t &) noexcept
- void **operator delete** (void \*, std::align\_val\_t) noexcept
- void **operator delete** (void \*, std::align\_val\_t, const std::nothrow\_t &) noexcept
- void **operator delete** (void \*, std::size\_t) noexcept
- void **operator delete** (void \*, std::size\_t, std::align\_val\_t) noexcept

- void **operator delete** (void \*, void \*) noexcept
- void **operator delete[]** (void \*) noexcept
- void **operator delete[]** (void \*, const std::nothrow\_t &) noexcept
- void **operator delete[]** (void \*, std::align\_val\_t) noexcept
- void **operator delete[]** (void \*, std::align\_val\_t, const std::nothrow\_t &) noexcept
- void **operator delete[]** (void \*, std::size\_t) noexcept
- void **operator delete[]** (void \*, std::size\_t, std::align\_val\_t) noexcept
- void **operator delete[]** (void \*, void \*) noexcept
- void \* **operator new** (std::size\_t)
- void \* **operator new** (std::size\_t, const std::nothrow\_t &) noexcept
- void \* **operator new** (std::size\_t, std::align\_val\_t)
- void \* **operator new** (std::size\_t, std::align\_val\_t, const std::nothrow\_t &) noexcept
- void \* **operator new** (std::size\_t, void \*\_\_p) noexcept
- void \* **operator new[]** (std::size\_t)
- void \* **operator new[]** (std::size\_t, const std::nothrow\_t &) noexcept
- void \* **operator new[]** (std::size\_t, std::align\_val\_t)
- void \* **operator new[]** (std::size\_t, std::align\_val\_t, const std::nothrow\_t &) noexcept
- void \* **operator new[]** (std::size\_t, void \*\_\_p) noexcept
- **new\_handler** std::set\_new\_handler (**new\_handler**) throw ()

## Variables

- constexpr **destroying\_delete\_t** std::destroying\_delete
- const nothrow\_t **std::nothrow**

### 6.728.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see [https://gcc.gnu.org/onlinedocs/libstdc++/manual/dynamic\\_memory.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/dynamic_memory.html) for more.

### 6.728.2 Function Documentation

#### **operator new()**

```
void * operator new (
    std::size_t ) [nodiscard]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

## 6.729 typeinfo File Reference

### Classes

- class **std::bad\_cast**
- class **std::bad\_typeid**
- class **std::type\_info**



**Namespaces**

- namespace [std](#)

**Macros**

- `#define __glibcxx_want_constexpr_typeinfo`
- `#define __GXX_MERGED_TYPEINFO_NAMES`
- `#define __GXX_TYPEINFO_EQUALITY_INLINE`
- `#define _TYPEINFO`

**6.729.1 Detailed Description**

This is a Standard C++ Library header.

## Index

- `_AlgorithmStrategy`
  - `__gnu_parallel`, [423](#)
- `_BALLOC_ALIGN_BYTES`
  - `bitmap_allocator.h`, [3606](#)
- `_BinIndex`
  - `__gnu_parallel`, [422](#)
- `_Bit_scan_forward`
  - `__gnu_cxx`, [394](#)
- `_CASable`
  - `__gnu_parallel`, [422](#)
- `_CASable_bits`
  - `__gnu_parallel`, [462](#)
- `_CASable_mask`
  - `__gnu_parallel`, [462](#)
- `_Construct`
  - `std`, [587](#)
- `_DRandomShufflingGlobalData`
  - `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter >`, [872](#)
- `_Destroy`
  - `std`, [587](#)
- `_Destroy_n`
  - `std`, [587](#)
- `_Distance_precision`
  - `__gnu_debug`, [412](#)
- `_FindAlgorithm`
  - `__gnu_parallel`, [423](#)
- `_Find_first`
  - `SGI`, [152](#)
- `_Find_next`
  - `SGI`, [152](#)
- `_GLIBCXX_BAL_QUICKSORT`
  - `features.h`, [3728](#)
- `_GLIBCXX_CALL`
  - `compiletime_settings.h`, [3726](#)
- `_GLIBCXX_DEBUG_VERIFY_AT_F`
  - `macros.h`, [3583](#)
- `_GLIBCXX_DEQUE_BUF_SIZE`
  - `stl_deque.h`, [3483](#)
- `_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
  - `features.h`, [3728](#)
- `_GLIBCXX_FIND_EQUAL_SPLIT`
  - `features.h`, [3728](#)
- `_GLIBCXX_FIND_GROWING_BLOCKS`
  - `features.h`, [3728](#)
- `_GLIBCXX_MERGESORT`
  - `features.h`, [3729](#)
- `_GLIBCXX_PARALLEL_ASSERTIONS`
  - `compiletime_settings.h`, [3727](#)
- `_GLIBCXX_PARALLEL_CONDITION`
  - `settings.h`, [3744](#)
- `_GLIBCXX_PARALLEL_LENGTH`
  - `multiway_merge.h`, [3736](#)
- `_GLIBCXX_QUICKSORT`
  - `features.h`, [3729](#)
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
  - `compiletime_settings.h`, [3727](#)
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
  - `compiletime_settings.h`, [3727](#)
- `_GLIBCXX_SCALE_DOWN_FPU`
  - `compiletime_settings.h`, [3727](#)
- `_GLIBCXX_TREE_DYNAMIC_BALANCING`
  - `features.h`, [3729](#)
- `_GLIBCXX_TREE_FULL_COPY`
  - `features.h`, [3729](#)
- `_GLIBCXX_TREE_INITIAL_SPLITTING`
  - `features.h`, [3729](#)
- `_GLIBCXX_VERBOSE_LEVEL`
  - `compiletime_settings.h`, [3727](#)
- `_GLIBCXX_VOLATILE`
  - `partition.h`, [3740](#)
  - `queue.h`, [3741](#)
- `_GuardedIterator`
  - `__gnu_parallel::_GuardedIterator<_RAIter, _Compare >`, [886](#)
- `_LoserTreeBase`
  - `__gnu_parallel::_LoserTreeBase<_Tp, _Compare >`, [918](#)
- `_M_allocate_and_copy`
  - `std::vector<_Tp, _Alloc >`, [3294](#)
  - `std::vector<bool, _Alloc >`, [3315](#)
- `_M_allocate_single_object`
  - `__gnu_cxx::bitmap_allocator<_Tp >`, [1758](#)
- `_M_attach`
  - `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >`, [963](#)
  - `__gnu_debug::_Safe_iterator_base`, [972](#)
  - `__gnu_debug::_Safe_local_iterator<_Iterator, _UContainer >`, [978](#)
  - `__gnu_debug::_Safe_local_iterator_base`, [986](#)
- `_M_attach_single`
  - `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >`, [964](#)
  - `__gnu_debug::_Safe_iterator_base`, [972](#)
  - `__gnu_debug::_Safe_local_iterator<_Iterator, _UContainer >`, [978](#), [979](#)
  - `__gnu_debug::_Safe_local_iterator_base`, [986](#)
- `_M_attached_to`
  - `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >`, [964](#)
  - `__gnu_debug::_Safe_iterator_base`, [972](#)

- \_\_gnu\_debug::Safe\_local\_iterator< \_\_Iterator, \_\_UContainer >, 979
- \_\_gnu\_debug::Safe\_local\_iterator\_base, 986
- \_M\_before\_dereferenceable
  - \_\_gnu\_debug::Safe\_iterator< \_\_Iterator, \_\_Sequence, \_\_Category >, 964
- \_M\_begin
  - \_\_gnu\_parallel::Piece< \_\_DifferenceTp >, 942
- \_M\_bin\_proc
  - \_\_gnu\_parallel::DRandomShufflingGlobalData< \_\_RAIter >, 872
- \_M\_bins\_begin
  - \_\_gnu\_parallel::DRSSorterPU< \_\_RAIter, \_\_RandomNumberGenerator >, 874
- \_M\_buf
  - \_\_gnu\_cxx::enc\_filebuf< \_\_CharT >, 2099
  - \_\_gnu\_cxx::stdio\_filebuf< \_\_CharT, \_\_Traits >, 3006
  - std::basic\_filebuf< \_\_CharT, \_\_Traits >, 1108
- \_M\_buf\_locale
  - \_\_gnu\_cxx::enc\_filebuf< \_\_CharT >, 2099
  - \_\_gnu\_cxx::stdio\_filebuf< \_\_CharT, \_\_Traits >, 3006
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_\_CharT, \_\_Traits >, 3024
  - std::basic\_filebuf< \_\_CharT, \_\_Traits >, 1108
  - std::basic\_streambuf< \_\_CharT, \_\_Traits >, 1472
  - std::basic\_stringbuf< \_\_CharT, \_\_Traits, \_\_Alloc >, 1667
  - std::wbuffer\_convert< \_\_Codecvt, \_\_Elem, \_\_Tr >, 3343
- \_M\_buf\_size
  - \_\_gnu\_cxx::enc\_filebuf< \_\_CharT >, 2099
  - \_\_gnu\_cxx::stdio\_filebuf< \_\_CharT, \_\_Traits >, 3006
  - std::basic\_filebuf< \_\_CharT, \_\_Traits >, 1108
- \_M\_can\_compare
  - \_\_gnu\_debug::Safe\_iterator< \_\_Iterator, \_\_Sequence, \_\_Category >, 964
  - \_\_gnu\_debug::Safe\_iterator\_base, 972
  - \_\_gnu\_debug::Safe\_local\_iterator< \_\_Iterator, \_\_UContainer >, 979
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 986
- \_M\_clear
  - \_\_gnu\_cxx::free\_list, 2161
- \_M\_comp
  - \_\_gnu\_parallel::LoserTreeBase< \_\_Tp, \_\_Compare >, 919
- \_M\_const\_iterators
  - \_\_gnu\_debug::Safe\_forward\_list< \_\_SafeSequence >, 959
  - \_\_gnu\_debug::Safe\_node\_sequence< \_\_Sequence >, 991
  - \_\_gnu\_debug::Safe\_sequence< \_\_Sequence >, 994
  - \_\_gnu\_debug::Safe\_sequence\_base, 997
  - \_\_gnu\_debug::Safe\_unordered\_container< \_\_Container >, 1000
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 1004
- \_M\_const\_local\_iterators
  - \_\_gnu\_debug::Safe\_unordered\_container< \_\_Container >, 1000
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 1004
- \_M\_create\_node
  - std::list< \_\_Tp, \_\_Alloc >, 2333
- \_M\_create\_pback
  - \_\_gnu\_cxx::enc\_filebuf< \_\_CharT >, 2086
  - \_\_gnu\_cxx::stdio\_filebuf< \_\_CharT, \_\_Traits >, 2991
  - std::basic\_filebuf< \_\_CharT, \_\_Traits >, 1093
- \_M\_deallocate\_single\_object
  - \_\_gnu\_cxx::bitmap\_allocator< \_\_Tp >, 1758
- \_M\_dereferenceable
  - \_\_gnu\_debug::Safe\_iterator< \_\_Iterator, \_\_Sequence, \_\_Category >, 964
  - \_\_gnu\_debug::Safe\_local\_iterator< \_\_Iterator, \_\_UContainer >, 979
- \_M\_destroy\_pback
  - \_\_gnu\_cxx::enc\_filebuf< \_\_CharT >, 2086
  - \_\_gnu\_cxx::stdio\_filebuf< \_\_CharT, \_\_Traits >, 2991
  - std::basic\_filebuf< \_\_CharT, \_\_Traits >, 1093
- \_M\_detach
  - \_\_gnu\_debug::Safe\_iterator< \_\_Iterator, \_\_Sequence, \_\_Category >, 965
  - \_\_gnu\_debug::Safe\_iterator\_base, 972
  - \_\_gnu\_debug::Safe\_local\_iterator< \_\_Iterator, \_\_UContainer >, 979
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 987
- \_M\_detach\_all
  - \_\_gnu\_debug::Safe\_forward\_list< \_\_SafeSequence >, 959
  - \_\_gnu\_debug::Safe\_node\_sequence< \_\_Sequence >, 990
  - \_\_gnu\_debug::Safe\_sequence< \_\_Sequence >, 993
  - \_\_gnu\_debug::Safe\_sequence\_base, 996
  - \_\_gnu\_debug::Safe\_unordered\_container< \_\_Container >, 999
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 1003
- \_M\_detach\_single
  - \_\_gnu\_debug::Safe\_iterator< \_\_Iterator, \_\_Sequence, \_\_Category >, 965
  - \_\_gnu\_debug::Safe\_iterator\_base, 972
  - \_\_gnu\_debug::Safe\_local\_iterator< \_\_Iterator, \_\_UContainer >, 979
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 987
- \_M\_detach\_singular
  - \_\_gnu\_debug::Safe\_forward\_list< \_\_SafeSequence >, 959
  - \_\_gnu\_debug::Safe\_node\_sequence< \_\_Sequence >, 990

- `__gnu_debug::__Safe_sequence< _Sequence >`, 993
- `__gnu_debug::__Safe_sequence_base`, 996
- `__gnu_debug::__Safe_unordered_container< _Container >`, 999
- `__gnu_debug::__Safe_unordered_container_base`, 1003
- `_M_dist`
  - `__gnu_parallel::__DRandomShufflingGlobalData< _RAIter >`, 872
- `_M_elements_leftover`
  - `__gnu_parallel::__QSBThreadLocal< _RAIter >`, 950
- `_M_end`
  - `__gnu_parallel::__Piece< _DifferenceTp >`, 942
- `_M_ext_buf`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2100
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3006
  - `std::basic_filebuf< _CharT, _Traits >`, 1108
- `_M_ext_buf_size`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2100
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3007
  - `std::basic_filebuf< _CharT, _Traits >`, 1108
- `_M_ext_next`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2100
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3007
  - `std::basic_filebuf< _CharT, _Traits >`, 1108
- `_M_fill_initialize`
  - `std::deque< _Tp, _Alloc >`, 2036
- `_M_finish_iterator`
  - `__gnu_parallel::__accumulate_selector< _It >`, 730
  - `__gnu_parallel::__adjacent_difference_selector< _It >`, 731
  - `__gnu_parallel::__count_if_selector< _It, _Diff >`, 747
  - `__gnu_parallel::__count_selector< _It, _Diff >`, 749
  - `__gnu_parallel::__fill_selector< _It >`, 765
  - `__gnu_parallel::__for_each_selector< _It >`, 769
  - `__gnu_parallel::__generate_selector< _It >`, 771
  - `__gnu_parallel::__generic_for_each_selector< _It >`, 773
  - `__gnu_parallel::__identity_selector< _It >`, 774
  - `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`, 776
  - `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >`, 801
  - `__gnu_parallel::__replace_selector< _It, _Tp >`, 803
  - `__gnu_parallel::__transform1_selector< _It >`, 805
  - `__gnu_parallel::__transform2_selector< _It >`, 806
- `_M_first`
  - `__gnu_parallel::__Job< _DifferenceTp >`, 901
- `_M_first_insert`
  - `__gnu_parallel::__LoserTreeBase< _Tp, _Compare >`, 919
- `_M_gcount`
  - `std::basic_fstream< _CharT, _Traits >`, 1155
  - `std::basic_ifstream< _CharT, _Traits >`, 1198
  - `std::basic_iostream< _CharT, _Traits >`, 1267
  - `std::basic_istream< _CharT, _Traits >`, 1304
  - `std::basic_istream< _CharT, _Traits >::sentry`, 2893
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1344
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1712
- `_M_get`
  - `__gnu_cxx::free_list`, 2161
- `_M_get_mutex`
  - `__gnu_debug::__Safe_forward_list< _SafeSequence >`, 959
  - `__gnu_debug::__Safe_iterator< _Iterator, _Sequence, _Category >`, 965
  - `__gnu_debug::__Safe_iterator_base`, 973
  - `__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >`, 979
  - `__gnu_debug::__Safe_local_iterator_base`, 987
  - `__gnu_debug::__Safe_node_sequence< _Sequence >`, 990
  - `__gnu_debug::__Safe_sequence< _Sequence >`, 993
  - `__gnu_debug::__Safe_sequence_base`, 996
  - `__gnu_debug::__Safe_unordered_container< _Container >`, 999
  - `__gnu_debug::__Safe_unordered_container_base`, 1003
- `_M_get_result`
  - `std::__basic_future< _Res >`, 740
  - `std::future< _Res >`, 2174
  - `std::future< _Res & >`, 2176
  - `std::future< void >`, 2178
  - `std::shared_future< _Res >`, 2952
  - `std::shared_future< _Res & >`, 2954
  - `std::shared_future< void >`, 2957
- `_M_getloc`
  - `std::basic_fstream< _CharT, _Traits >`, 1120
  - `std::basic_ifstream< _CharT, _Traits >`, 1171
  - `std::basic_ios< _CharT, _Traits >`, 1210
  - `std::basic_iostream< _CharT, _Traits >`, 1233
  - `std::basic_istream< _CharT, _Traits >`, 1278
  - `std::basic_istream< _CharT, _Traits >::sentry`, 2870
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1318
  - `std::basic_ofstream< _CharT, _Traits >`, 1356
  - `std::basic_ostream< _CharT, _Traits >`, 1388
  - `std::basic_ostream< _CharT, _Traits >::sentry`, 2903
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1421
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1678
  - `std::ios_base`, 2261

- `_M_global`
  - `__gnu_parallel::QSBThreadLocal<_RAIter>`, 950
- `_M_in_beg`
  - `__gnu_cxx::enc_filebuf<_CharT>`, 2100
  - `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`, 3007
  - `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, 3024
  - `std::basic_filebuf<_CharT, _Traits>`, 1108
  - `std::basic_streambuf<_CharT, _Traits>`, 1472
  - `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, 1667
  - `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`, 3343
- `_M_in_cur`
  - `__gnu_cxx::enc_filebuf<_CharT>`, 2100
  - `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`, 3007
  - `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, 3025
  - `std::basic_filebuf<_CharT, _Traits>`, 1108
  - `std::basic_streambuf<_CharT, _Traits>`, 1473
  - `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, 1667
  - `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`, 3343
- `_M_in_end`
  - `__gnu_cxx::enc_filebuf<_CharT>`, 2100
  - `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`, 3007
  - `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, 3025
  - `std::basic_filebuf<_CharT, _Traits>`, 1109
  - `std::basic_streambuf<_CharT, _Traits>`, 1473
  - `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, 1668
  - `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`, 3343
- `_M_in_same_bucket`
  - `__gnu_debug::Safe_local_iterator<_Iterator, _UContainer>`, 979
- `_M_incrementable`
  - `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category>`, 965
  - `__gnu_debug::Safe_local_iterator<_Iterator, _UContainer>`, 980
- `_M_initial`
  - `__gnu_parallel::QSBThreadLocal<_RAIter>`, 950
- `_M_initialize_map`
  - `std::Deque_base<_Tp, _Alloc>`, 869
- `_M_insert`
  - `__gnu_cxx::free_list`, 2161
- `_M_invalidate`
  - `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category>`, 965
  - `__gnu_debug::Safe_iterator_base`, 973
  - `__gnu_debug::Safe_local_iterator<_Iterator, _UContainer>`, 980
  - `__gnu_debug::Safe_local_iterator_base`, 987
- `_M_invalidate_all`
  - `__gnu_debug::Safe_sequence<_Sequence>`, 993
  - `__gnu_debug::Safe_sequence_base`, 996
- `__gnu_debug::Safe_unordered_container<_Container>`, 999
- `__gnu_debug::Safe_unordered_container_base`, 1004
- `_M_invalidate_if`
  - `__gnu_debug::Safe_forward_list<_SafeSequence>`, 959
  - `__gnu_debug::Safe_node_sequence<_Sequence>`, 990
  - `__gnu_debug::Safe_sequence<_Sequence>`, 993
  - `__gnu_debug::Safe_unordered_container<_Container>`, 999
- `_M_invalidate_local_if`
  - `__gnu_debug::Safe_unordered_container<_Container>`, 1000
- `_M_is_before_begin`
  - `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category>`, 965
- `_M_is_begin`
  - `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category>`, 965
  - `__gnu_debug::Safe_local_iterator<_Iterator, _UContainer>`, 980
- `_M_is_beginnest`
  - `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category>`, 966
- `_M_is_end`
  - `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category>`, 966
  - `__gnu_debug::Safe_local_iterator<_Iterator, _UContainer>`, 980
- `_M_iterators`
  - `__gnu_debug::Safe_forward_list<_SafeSequence>`, 959
  - `__gnu_debug::Safe_node_sequence<_Sequence>`, 991
  - `__gnu_debug::Safe_sequence<_Sequence>`, 994
  - `__gnu_debug::Safe_sequence_base`, 997
  - `__gnu_debug::Safe_unordered_container<_Container>`, 1000
  - `__gnu_debug::Safe_unordered_container_base`, 1004
- `_M_key`
  - `__gnu_parallel::LoserTreeBase<_Tp, _Compare>::Loser`, 912
- `_M_last`
  - `__gnu_parallel::Job<_DifferenceTp>`, 901
- `_M_leftover_parts`
  - `__gnu_parallel::QSBThreadLocal<_RAIter>`, 950
- `_M_load`
  - `__gnu_parallel::Job<_DifferenceTp>`, 901
- `_M_local_iterators`

- `__gnu_debug::Safe_unordered_container< _Container >`, 1001
  - `__gnu_debug::Safe_unordered_container_base`, 1004
- `_M_log_k`
  - `__gnu_parallel::LoserTree< __stable, _Tp, _Compare >`, 915
  - `__gnu_parallel::LoserTree< false, _Tp, _Compare >`, 917
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >`, 919
- `_M_losers`
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >`, 919
- `_M_mode`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2100
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3007
  - `std::basic_filebuf< _CharT, _Traits >`, 1109
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1668
- `_M_new_elements_at_back`
  - `std::deque< _Tp, _Alloc >`, 2036
- `_M_new_elements_at_front`
  - `std::deque< _Tp, _Alloc >`, 2036
- `_M_next`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 968
  - `__gnu_debug::Safe_iterator_base`, 973
  - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 983
  - `__gnu_debug::Safe_local_iterator_base`, 988
- `_M_num_bins`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >`, 872
- `_M_num_bits`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >`, 872
- `_M_num_threads`
  - `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >`, 874
  - `__gnu_parallel::PMWMSSortingData< _RAIter >`, 944
  - `__gnu_parallel::QSBThreadLocal< _RAIter >`, 950
- `_M_offsets`
  - `__gnu_parallel::PMWMSSortingData< _RAIter >`, 944
- `_M_out_beg`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2100
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3007
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3025
  - `std::basic_filebuf< _CharT, _Traits >`, 1109
  - `std::basic_streambuf< _CharT, _Traits >`, 1473
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1668
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3343
- `_M_out_cur`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2100
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3007
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3025
  - `std::basic_filebuf< _CharT, _Traits >`, 1109
  - `std::basic_streambuf< _CharT, _Traits >`, 1473
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1668
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3343
- `_M_out_end`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2101
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3007
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3025
  - `std::basic_filebuf< _CharT, _Traits >`, 1109
  - `std::basic_streambuf< _CharT, _Traits >`, 1473
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1668
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3343
- `_M_pback`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2101
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3008
  - `std::basic_filebuf< _CharT, _Traits >`, 1109
- `_M_pback_cur_save`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2101
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3008
  - `std::basic_filebuf< _CharT, _Traits >`, 1109
- `_M_pback_end_save`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2101
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3008
  - `std::basic_filebuf< _CharT, _Traits >`, 1110
- `_M_pback_init`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2101
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3008
  - `std::basic_filebuf< _CharT, _Traits >`, 1110
- `_M_pieces`
  - `__gnu_parallel::PMWMSSortingData< _RAIter >`, 944
- `_M_pop_back_aux`
  - `std::deque< _Tp, _Alloc >`, 2037
- `_M_pop_front_aux`
  - `std::deque< _Tp, _Alloc >`, 2037
- `_M_prior`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 968
  - `__gnu_debug::Safe_iterator_base`, 973
  - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 983
  - `__gnu_debug::Safe_local_iterator_base`, 988
- `_M_push_back_aux`
  - `std::deque< _Tp, _Alloc >`, 2037
- `_M_push_front_aux`
  - `std::deque< _Tp, _Alloc >`, 2037
- `_M_range_check`
  - `std::deque< _Tp, _Alloc >`, 2037

- std::vector< \_Tp, \_Alloc >, 3294
- std::vector< bool, \_Alloc >, 3315
- \_M\_range\_initialize
  - std::deque< \_Tp, \_Alloc >, 2037, 2038
- \_M\_reading
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2101
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3008
  - std::basic\_filebuf< \_CharT, \_Traits >, 1110
- \_M\_reallocate\_map
  - std::deque< \_Tp, \_Alloc >, 2038
- \_M\_requested\_size
  - \_\_gnu\_cxx::Temporary\_buffer< \_ForwardIterator, \_Tp >, 1019
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 3048
  - std::Temporary\_buffer< \_ForwardIterator, \_Tp >, 1021
- \_M\_reserve\_elements\_at\_back
  - std::deque< \_Tp, \_Alloc >, 2038
- \_M\_reserve\_elements\_at\_front
  - std::deque< \_Tp, \_Alloc >, 2039
- \_M\_reserve\_map\_at\_back
  - std::deque< \_Tp, \_Alloc >, 2039
- \_M\_reserve\_map\_at\_front
  - std::deque< \_Tp, \_Alloc >, 2039
- \_M\_reset
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 966
  - \_\_gnu\_debug::Safe\_iterator\_base, 973
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_UContainer >, 980
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 987
- \_M\_revalidate\_singular
  - \_\_gnu\_debug::Safe\_forward\_list< \_SafeSequence >, 959
  - \_\_gnu\_debug::Safe\_node\_sequence< \_Sequence >, 990
  - \_\_gnu\_debug::Safe\_sequence< \_Sequence >, 993
  - \_\_gnu\_debug::Safe\_sequence\_base, 997
  - \_\_gnu\_debug::Safe\_unordered\_container< \_Container >, 1000
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 1004
- \_M\_samples
  - \_\_gnu\_parallel::PMWMSSortingData< \_RAIter >, 944
- \_M\_sd
  - \_\_gnu\_parallel::DRSSorterPU< \_RAIter, \_RandomNumberGenerator >, 874
- \_M\_seed
  - \_\_gnu\_parallel::DRSSorterPU< \_RAIter, \_RandomNumberGenerator >, 874
- \_M\_sequence
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 969
  - \_\_gnu\_debug::Safe\_iterator\_base, 974
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_UContainer >, 983
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 988
- \_M\_sequential\_algorithm
  - \_\_gnu\_parallel::adjacent\_find\_selector, 732
  - \_\_gnu\_parallel::find\_first\_of\_selector< \_FIterator >, 766
  - \_\_gnu\_parallel::find\_if\_selector, 767
  - \_\_gnu\_parallel::mismatch\_selector, 779
- \_M\_set\_buffer
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2087
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2991
  - std::basic\_filebuf< \_CharT, \_Traits >, 1093
- \_M\_set\_node
  - std::Deque\_iterator< \_Tp, \_Ref, \_Ptr >, 871
- \_M\_singular
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 966
  - \_\_gnu\_debug::Safe\_iterator\_base, 973
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_UContainer >, 980
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 987
- \_M\_source
  - \_\_gnu\_parallel::DRandomShufflingGlobalData< \_RAIter >, 873
  - \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >::\_Loser, 912
  - \_\_gnu\_parallel::PMWMSSortingData< \_RAIter >, 945
- \_M\_starts
  - \_\_gnu\_parallel::DRandomShufflingGlobalData< \_RAIter >, 873
  - \_\_gnu\_parallel::PMWMSSortingData< \_RAIter >, 945
- \_M\_sup
  - \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >::\_Loser, 912
- \_M\_swap
  - \_\_gnu\_debug::Safe\_forward\_list< \_SafeSequence >, 959
  - \_\_gnu\_debug::Safe\_node\_sequence< \_Sequence >, 990
  - \_\_gnu\_debug::Safe\_sequence< \_Sequence >, 993
  - \_\_gnu\_debug::Safe\_sequence\_base, 997
  - \_\_gnu\_debug::Safe\_unordered\_container< \_Container >, 1000
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 1004
- \_M\_temporaries
  - \_\_gnu\_parallel::DRandomShufflingGlobalData<



- [\\_RAIter >, 873](#)
- [\\_M\\_temporary](#)
  - [\\_\\_gnu\\_parallel::PMWMSortingData< \\_RAIter >, 945](#)
- [\\_M\\_transfer\\_from\\_if](#)
  - [\\_\\_gnu\\_debug::Safe\\_forward\\_list< \\_SafeSequence >, 959](#)
  - [\\_\\_gnu\\_debug::Safe\\_node\\_sequence< \\_Sequence >, 990](#)
  - [\\_\\_gnu\\_debug::Safe\\_sequence< \\_Sequence >, 994](#)
- [\\_M\\_unlink](#)
  - [\\_\\_gnu\\_debug::Safe\\_iterator< \\_Iterator, \\_Sequence, \\_Category >, 966](#)
  - [\\_\\_gnu\\_debug::Safe\\_iterator\\_base, 973](#)
  - [\\_\\_gnu\\_debug::Safe\\_local\\_iterator< \\_Iterator, \\_UContainer >, 980](#)
  - [\\_\\_gnu\\_debug::Safe\\_local\\_iterator\\_base, 987](#)
- [\\_M\\_use\\_pointer](#)
  - [\\_\\_gnu\\_parallel::LoserTreeTraits< \\_Tp >, 926](#)
- [\\_M\\_value\\_initialized](#)
  - [\\_\\_gnu\\_debug::Safe\\_iterator< \\_Iterator, \\_Sequence, \\_Category >, 966](#)
  - [\\_\\_gnu\\_debug::Safe\\_local\\_iterator< \\_Iterator, \\_UContainer >, 981](#)
- [\\_M\\_version](#)
  - [\\_\\_gnu\\_debug::Safe\\_forward\\_list< \\_SafeSequence >, 960](#)
  - [\\_\\_gnu\\_debug::Safe\\_iterator< \\_Iterator, \\_Sequence, \\_Category >, 969](#)
  - [\\_\\_gnu\\_debug::Safe\\_iterator\\_base, 974](#)
  - [\\_\\_gnu\\_debug::Safe\\_local\\_iterator< \\_Iterator, \\_UContainer >, 983](#)
  - [\\_\\_gnu\\_debug::Safe\\_local\\_iterator\\_base, 988](#)
  - [\\_\\_gnu\\_debug::Safe\\_node\\_sequence< \\_Sequence >, 991](#)
  - [\\_\\_gnu\\_debug::Safe\\_sequence< \\_Sequence >, 994](#)
  - [\\_\\_gnu\\_debug::Safe\\_sequence\\_base, 997](#)
  - [\\_\\_gnu\\_debug::Safe\\_unordered\\_container< \\_Container >, 1001](#)
  - [\\_\\_gnu\\_debug::Safe\\_unordered\\_container\\_base, 1005](#)
- [\\_M\\_w](#)
  - [std::\\_Base\\_bitset< 0 >, 861](#)
  - [std::\\_Base\\_bitset< 1 >, 863](#)
  - [std::\\_Base\\_bitset< \\_Nw >, 859](#)
  - [std::tr2::\\_\\_dynamic\\_bitset\\_base< \\_WordT, \\_Alloc >, 764](#)
- [\\_MultiwayMergeAlgorithm](#)
  - [\\_\\_gnu\\_parallel, 423](#)
- [\\_Opcode](#)
  - [Base and Implementation Classes, 272](#)
- [\\_Parallelism](#)
  - [\\_\\_gnu\\_parallel, 423](#)
- [\\_PartialSumAlgorithm](#)
  - [\\_\\_gnu\\_parallel, 423](#)
- [\\_Piece](#)
  - [\\_\\_gnu\\_parallel::QSBThreadLocal< \\_RAIter >, 950](#)
- [\\_PseudoSequence](#)
  - [\\_\\_gnu\\_parallel::PseudoSequence< \\_Tp, \\_DifferenceTp >, 948](#)
- [\\_QSBThreadLocal](#)
  - [\\_\\_gnu\\_parallel::QSBThreadLocal< \\_RAIter >, 950](#)
- [\\_RandomNumber](#)
  - [\\_\\_gnu\\_parallel::RandomNumber, 951](#)
- [\\_RestrictedBoundedConcurrentQueue](#)
  - [\\_\\_gnu\\_parallel::RestrictedBoundedConcurrentQueue< \\_Tp >, 955](#)
- [\\_S\\_constant](#)
  - [\\_\\_gnu\\_debug::Safe\\_iterator< \\_Iterator, \\_Sequence, \\_Category >, 967](#)
  - [\\_\\_gnu\\_debug::Safe\\_local\\_iterator< \\_Iterator, \\_UContainer >, 981](#)
- [\\_Safe\\_iterator](#)
  - [\\_\\_gnu\\_debug::Safe\\_iterator< \\_Iterator, \\_Sequence, \\_Category >, 962, 963](#)
- [\\_Safe\\_iterator\\_base](#)
  - [\\_\\_gnu\\_debug::Safe\\_iterator\\_base, 971](#)
- [\\_Safe\\_local\\_iterator](#)
  - [\\_\\_gnu\\_debug::Safe\\_local\\_iterator< \\_Iterator, \\_UContainer >, 977](#)
- [\\_Safe\\_local\\_iterator\\_base](#)
  - [\\_\\_gnu\\_debug::Safe\\_local\\_iterator\\_base, 985](#)
- [\\_SequenceIndex](#)
  - [\\_\\_gnu\\_parallel, 422](#)
- [\\_SortAlgorithm](#)
  - [\\_\\_gnu\\_parallel, 423](#)
- [\\_SplittingAlgorithm](#)
  - [\\_\\_gnu\\_parallel, 423](#)
- [\\_Temporary\\_buffer](#)
  - [\\_\\_gnu\\_cxx::Temporary\\_buffer< \\_ForwardIterator, \\_Tp >, 1019](#)
  - [std::Temporary\\_buffer< \\_ForwardIterator, \\_Tp >, 1021](#)
- [\\_ThreadIndex](#)
  - [\\_\\_gnu\\_parallel, 422](#)
- [\\_TokenT](#)
  - [std::\\_\\_detail::Scanner< \\_CharT >, 1008](#)
- [\\_Unchecked\\_flip](#)
  - [SGI, 152](#)
- [\\_Unchecked\\_reset](#)
  - [SGI, 152](#)
- [\\_Unchecked\\_set](#)
  - [SGI, 153](#)
- [\\_Unchecked\\_test](#)
  - [SGI, 153](#)
- [\\_\\_addressof](#)



- Utilities, 298
- \_\_allocator\_base
  - Allocators, 324
- \_\_attribute
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2385
- \_\_base
  - \_\_gnu\_debug, 412
- \_\_begin1\_iterator
  - \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp >, 776
- \_\_begin2\_iterator
  - \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp >, 776
- \_\_bins\_end
  - \_\_gnu\_parallel::\_\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >, 873
- \_\_bit\_allocate
  - \_\_gnu\_cxx::\_\_detail, 404
- \_\_bit\_free
  - \_\_gnu\_cxx::\_\_detail, 404
- \_\_calc\_borders
  - \_\_gnu\_parallel, 423
- \_\_check\_singular
  - \_\_gnu\_debug, 412
- \_\_check\_singular\_aux
  - \_\_gnu\_debug, 412
- \_\_check\_string
  - \_\_gnu\_debug, 412
- \_\_compare\_and\_swap
  - \_\_gnu\_parallel, 424
- \_\_cpp\_lib\_experimental\_detect
  - Detection idiom, 285
- \_\_ctype\_type
  - std::basic\_ios< \_CharT, \_Traits >, 1207
- \_\_cxa\_demangle
  - cxxabi.h, 3802
- \_\_cxxabiv1::\_\_forced\_unwind, 769
- \_\_decode2
  - \_\_gnu\_parallel, 424
- \_\_delete\_min\_insert
  - \_\_gnu\_parallel::\_\_LoserTree< \_\_stable, \_Tp, \_Compare >, 914
  - \_\_gnu\_parallel::\_\_LoserTree< false, \_Tp, \_Compare >, 916
- \_\_determine\_samples
  - \_\_gnu\_parallel, 424
- \_\_encode2
  - \_\_gnu\_parallel, 425
- \_\_equally\_split
  - \_\_gnu\_parallel, 425
- \_\_equally\_split\_point
  - \_\_gnu\_parallel, 426
- \_\_fetch\_and\_add
  - \_\_gnu\_parallel, 426
- \_\_find\_if\_not
  - std, 581
- \_\_find\_if\_not\_n
  - std, 581
- \_\_find\_template
  - \_\_gnu\_parallel, 426–428
- \_\_for\_each\_template\_random\_access
  - \_\_gnu\_parallel, 429
- \_\_for\_each\_template\_random\_access\_ed
  - \_\_gnu\_parallel, 430
- \_\_for\_each\_template\_random\_access\_omp\_loop
  - \_\_gnu\_parallel, 430
- \_\_for\_each\_template\_random\_access\_omp\_loop\_static
  - \_\_gnu\_parallel, 431
- \_\_for\_each\_template\_random\_access\_workstealing
  - \_\_gnu\_parallel, 431
- \_\_foreign\_iterator\_aux2
  - \_\_gnu\_debug, 413
- \_\_from\_chars\_alnum
  - std::\_\_detail, 656
- \_\_from\_chars\_pow2\_base
  - std::\_\_detail, 656
- \_\_gcd
  - std, 581
- \_\_gen\_two\_uniform\_ints
  - std, 581
- \_\_genrand\_bits
  - \_\_gnu\_parallel::\_\_RandomNumber, 952
- \_\_get\_distance
  - \_\_gnu\_debug, 413
- \_\_get\_min\_source
  - \_\_gnu\_parallel::\_\_LoserTree< \_\_stable, \_Tp, \_Compare >, 914
  - \_\_gnu\_parallel::\_\_LoserTree< false, \_Tp, \_Compare >, 916
  - \_\_gnu\_parallel::\_\_LoserTreeBase< \_Tp, \_Compare >, 919
- \_\_get\_num\_threads
  - \_\_gnu\_parallel::balanced\_quicksort\_tag, 1086
  - \_\_gnu\_parallel::balanced\_tag, 1087
  - \_\_gnu\_parallel::default\_parallel\_tag, 2023
  - \_\_gnu\_parallel::exact\_tag, 2117
  - \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, 2523
  - \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, 2524
  - \_\_gnu\_parallel::multiway\_mergesort\_tag, 2525
  - \_\_gnu\_parallel::omp\_loop\_static\_tag, 2682
  - \_\_gnu\_parallel::omp\_loop\_tag, 2683
  - \_\_gnu\_parallel::parallel\_tag, 2719
  - \_\_gnu\_parallel::quicksort\_tag, 2778
  - \_\_gnu\_parallel::sampling\_tag, 2857
  - \_\_gnu\_parallel::unbalanced\_tag, 3140
- \_\_glibcxx\_check\_erase

- macros.h, [3581](#)
- \_\_glibcxx\_check\_erase\_after
  - macros.h, [3581](#)
- \_\_glibcxx\_check\_erase\_range
  - macros.h, [3581](#)
- \_\_glibcxx\_check\_erase\_range\_after
  - macros.h, [3581](#)
- \_\_glibcxx\_check\_heap\_pred
  - macros.h, [3581](#)
- \_\_glibcxx\_check\_insert
  - macros.h, [3581](#)
- \_\_glibcxx\_check\_insert\_after
  - macros.h, [3581](#)
- \_\_glibcxx\_check\_insert\_range
  - macros.h, [3582](#)
- \_\_glibcxx\_check\_insert\_range\_after
  - macros.h, [3582](#)
- \_\_glibcxx\_check\_partitioned\_lower
  - macros.h, [3582](#)
- \_\_glibcxx\_check\_partitioned\_lower\_pred
  - macros.h, [3582](#)
- \_\_glibcxx\_check\_partitioned\_upper\_pred
  - macros.h, [3582](#)
- \_\_glibcxx\_check\_sorted\_pred
  - macros.h, [3583](#)
- \_\_gnu\_cxx, [377](#)
  - \_Bit\_scan\_forward, [394](#)
  - \_\_int\_traits, [394](#)
  - \_\_static\_pointer\_cast, [394](#)
  - iota, [394](#)
  - operator!=, [395](#)
  - operator<, [398](#)
  - operator<=, [399](#), [400](#)
  - operator>, [401](#), [402](#)
  - operator>=, [402](#), [403](#)
  - operator+, [396](#), [397](#)
  - operator==, [400](#), [401](#)
  - swap, [403](#)
- \_\_gnu\_cxx::Caster<\_ToType>, [865](#)
- \_\_gnu\_cxx::Char\_types<\_CharT>, [865](#)
- \_\_gnu\_cxx::ExtPtr\_allocator<\_Tp>, [878](#)
- \_\_gnu\_cxx::Invalid\_type, [891](#)
- \_\_gnu\_cxx::Pointer\_adapter<\_Storage\_policy>, [945](#)
- \_\_gnu\_cxx::Relative\_pointer\_impl<\_Tp>, [952](#)
- \_\_gnu\_cxx::Relative\_pointer\_impl<const \_Tp>, [954](#)
- \_\_gnu\_cxx::Std\_pointer\_impl<\_Tp>, [1017](#)
- \_\_gnu\_cxx::Temporary\_buffer<\_ForwardIterator, \_Tp>, [1018](#)
  - \_M\_requested\_size, [1019](#)
  - \_Temporary\_buffer, [1019](#)
  - begin, [1019](#)
  - end, [1019](#)
  - size, [1019](#)
- \_\_gnu\_cxx::Unqualified\_type<\_Tp>, [1021](#)
- \_\_gnu\_cxx::\_\_alloc\_traits<\_Alloc, typename>, [732](#)
  - allocate, [735](#), [736](#)
  - const\_void\_pointer, [734](#)
  - construct, [736](#)
  - deallocate, [737](#)
  - destroy, [737](#)
  - is\_always\_equal, [734](#)
  - max\_size, [738](#)
  - propagate\_on\_container\_copy\_assignment, [734](#)
  - propagate\_on\_container\_move\_assignment, [734](#)
  - propagate\_on\_container\_swap, [735](#)
  - select\_on\_container\_copy\_construction, [738](#)
  - void\_pointer, [735](#)
- \_\_gnu\_cxx::\_\_common\_pool\_policy<\_PoolTp, \_Thread>, [746](#)
- \_\_gnu\_cxx::\_\_detail, [404](#)
  - \_\_bit\_allocate, [404](#)
  - \_\_bit\_free, [404](#)
  - \_\_num\_bitmaps, [404](#)
  - \_\_num\_blocks, [405](#)
- \_\_gnu\_cxx::\_\_detail::Bitmap\_counter<\_Tp>, [863](#)
- \_\_gnu\_cxx::\_\_detail::Ffit\_finder<\_Tp>, [879](#)
- \_\_gnu\_cxx::\_\_detail::\_\_mini\_vector<\_Tp>, [777](#)
- \_\_gnu\_cxx::\_\_mt\_alloc<\_Tp, \_Poolp>, [780](#)
- \_\_gnu\_cxx::\_\_mt\_alloc\_base<\_Tp>, [781](#)
- \_\_gnu\_cxx::\_\_per\_type\_pool\_policy<\_Tp, \_PoolTp, \_Thread>, [790](#)
- \_\_gnu\_cxx::\_\_pool<\_Thread>, [791](#)
- \_\_gnu\_cxx::\_\_pool<false>, [791](#)
- \_\_gnu\_cxx::\_\_pool<true>, [792](#)
- \_\_gnu\_cxx::\_\_pool\_alloc<\_Tp>, [793](#)
- \_\_gnu\_cxx::\_\_pool\_alloc\_base, [794](#)
- \_\_gnu\_cxx::\_\_pool\_base, [795](#)
- \_\_gnu\_cxx::\_\_rc\_string\_base<\_CharT, \_Traits, \_Alloc>, [796](#)
- \_\_gnu\_cxx::\_\_scoped\_lock, [803](#)
- \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>, [807](#)
  - \_\_versa\_string, [811–813](#)
  - ~\_\_versa\_string, [814](#)
- append, [814–816](#)
- assign, [816–819](#)
- at, [819](#), [820](#)
- back, [820](#)
- begin, [821](#)
- c\_str, [821](#)
- capacity, [821](#)
- cbegin, [821](#)
- cend, [821](#)
- clear, [822](#)
- compare, [822–824](#)
- copy, [825](#)
- crbegin, [825](#)
- crend, [825](#)

data, 826  
 empty, 826  
 end, 826  
 erase, 826, 827  
 find, 827–829  
 find\_first\_not\_of, 829, 830  
 find\_first\_of, 831, 832  
 find\_last\_not\_of, 832–834  
 find\_last\_of, 834, 835  
 front, 836  
 get\_allocator, 836  
 insert, 836–840  
 length, 841  
 max\_size, 841  
 npos, 858  
 operator+=, 841, 843  
 operator=, 844, 845  
 operator[], 845  
 pop\_back, 846  
 push\_back, 846  
 rbegin, 846  
 rend, 847  
 replace, 847–853  
 reserve, 853  
 resize, 854  
 rfind, 855, 856  
 shrink\_to\_fit, 856  
 size, 856  
 substr, 857  
 swap, 857  
 \_\_gnu\_cxx::allocator< \_Tp >, 1027  
 \_\_gnu\_cxx::annotate\_base, 1063  
 \_\_gnu\_cxx::binary\_compose< \_Operation1, \_Operation2, \_Operation3 >, 1732  
   argument\_type, 1733  
   result\_type, 1733  
 \_\_gnu\_cxx::bitmap\_allocator< \_Tp >, 1757  
   \_M\_allocate\_single\_object, 1758  
   \_M\_deallocate\_single\_object, 1758  
 \_\_gnu\_cxx::char\_traits< \_CharT >, 1790  
 \_\_gnu\_cxx::character< \_Value, \_Int, \_St >, 1796  
 \_\_gnu\_cxx::condition\_base, 1877  
 \_\_gnu\_cxx::constant\_binary\_fun< \_Result, \_Arg1, \_Arg2 >, 1885  
 \_\_gnu\_cxx::constant\_unary\_fun< \_Result, \_Argument >, 1886  
 \_\_gnu\_cxx::constant\_void\_fun< \_Result >, 1887  
 \_\_gnu\_cxx::debug\_allocator< \_Alloc >, 2013  
 \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2083  
   \_M\_buf, 2099  
   \_M\_buf\_locale, 2099  
   \_M\_buf\_size, 2099  
   \_M\_create\_pback, 2086  
   \_M\_destroy\_pback, 2086  
   \_M\_ext\_buf, 2100  
   \_M\_ext\_buf\_size, 2100  
   \_M\_ext\_next, 2100  
   \_M\_in\_beg, 2100  
   \_M\_in\_cur, 2100  
   \_M\_in\_end, 2100  
   \_M\_mode, 2100  
   \_M\_out\_beg, 2100  
   \_M\_out\_cur, 2100  
   \_M\_out\_end, 2101  
   \_M\_pback, 2101  
   \_M\_pback\_cur\_save, 2101  
   \_M\_pback\_end\_save, 2101  
   \_M\_pback\_init, 2101  
   \_M\_reading, 2101  
   \_M\_set\_buffer, 2087  
 close, 2087  
 eback, 2087  
 egptr, 2087  
 ep\_ptr, 2087  
 gbump, 2088  
 getloc, 2088  
 gptr, 2088  
 imbue, 2088  
 in\_avail, 2089  
 is\_open, 2089  
 open, 2089, 2090  
 overflow, 2090  
 pbackfail, 2091  
 pbase, 2091  
 pbump, 2092  
 pptr, 2092  
 pubimbue, 2092  
 pubseekoff, 2092  
 pubseekpos, 2093  
 pubsetbuf, 2093  
 pubsync, 2093  
 sbumpc, 2093  
 seekoff, 2093  
 seekpos, 2094  
 setbuf, 2094  
 setg, 2094  
 setp, 2095  
 sgetc, 2095  
 sgetn, 2095  
 showmanyc, 2096  
 snextc, 2096  
 sputbackc, 2096  
 sputc, 2097  
 sputn, 2097  
 sungetc, 2097  
 sync, 2097  
 uflow, 2098  
 underflow, 2098

- xsgetn, 2098
- xspu, 2099
- \_\_gnu\_cxx::encoding\_char\_traits< \_CharT >, 2102
- \_\_gnu\_cxx::encoding\_state, 2103
- \_\_gnu\_cxx::equal\_to< \_Tp >, 2108
  - first\_argument\_type, 2108
  - result\_type, 2108
  - second\_argument\_type, 2108
- \_\_gnu\_cxx::forced\_error, 2135
  - what, 2136
- \_\_gnu\_cxx::free\_list, 2160
  - \_M\_clear, 2161
  - \_M\_get, 2161
  - \_M\_insert, 2161
- \_\_gnu\_cxx::hash\_map< \_Key, \_Tp, \_HashFn, \_EqualKey, \_Alloc >, 2227
- \_\_gnu\_cxx::hash\_multimap< \_Key, \_Tp, \_HashFn, \_EqualKey, \_Alloc >, 2229
- \_\_gnu\_cxx::hash\_multiset< \_Value, \_HashFcn, \_EqualKey, \_Alloc >, 2230
- \_\_gnu\_cxx::hash\_set< \_Value, \_HashFcn, \_EqualKey, \_Alloc >, 2232
- \_\_gnu\_cxx::limit\_condition, 2318
- \_\_gnu\_cxx::limit\_condition::always\_adjustor, 1062
- \_\_gnu\_cxx::limit\_condition::limit\_adjustor, 2318
- \_\_gnu\_cxx::limit\_condition::never\_adjustor, 2533
- \_\_gnu\_cxx::malloc\_allocator< \_Tp >, 2375
- \_\_gnu\_cxx::new\_allocator< \_Tp >, 2533
- \_\_gnu\_cxx::pair< \_T1, \_T2 >, 2707
  - first, 2710
  - first\_type, 2709
  - pair, 2709, 2710
  - second, 2710
  - second\_type, 2709
  - swap, 2710
- \_\_gnu\_cxx::project1st< \_Arg1, \_Arg2 >, 2769
  - first\_argument\_type, 2769
  - result\_type, 2769
  - second\_argument\_type, 2769
- \_\_gnu\_cxx::project2nd< \_Arg1, \_Arg2 >, 2769
  - first\_argument\_type, 2770
  - result\_type, 2770
  - second\_argument\_type, 2770
- \_\_gnu\_cxx::random\_condition, 2779
- \_\_gnu\_cxx::random\_condition::always\_adjustor, 1063
- \_\_gnu\_cxx::random\_condition::group\_adjustor, 2202
- \_\_gnu\_cxx::random\_condition::never\_adjustor, 2533
- \_\_gnu\_cxx::rb\_tree< \_Key, \_Value, \_KeyOfValue, \_Compare, \_Alloc >, 2796
- \_\_gnu\_cxx::recursive\_init\_error, 2810
  - what, 2810
- \_\_gnu\_cxx::rope< \_CharT, \_Alloc >, 2837
- \_\_gnu\_cxx::select1st< \_Pair >, 2860
  - argument\_type, 2861
  - result\_type, 2861
- \_\_gnu\_cxx::select2nd< \_Pair >, 2861
  - argument\_type, 2861
  - result\_type, 2861
- \_\_gnu\_cxx::slist< \_Tp, \_Alloc >, 2976
- \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2987
  - \_M\_buf, 3006
  - \_M\_buf\_locale, 3006
  - \_M\_buf\_size, 3006
  - \_M\_create\_pback, 2991
  - \_M\_destroy\_pback, 2991
  - \_M\_ext\_buf, 3006
  - \_M\_ext\_buf\_size, 3007
  - \_M\_ext\_next, 3007
  - \_M\_in\_beg, 3007
  - \_M\_in\_cur, 3007
  - \_M\_in\_end, 3007
  - \_M\_mode, 3007
  - \_M\_out\_beg, 3007
  - \_M\_out\_cur, 3007
  - \_M\_out\_end, 3007
  - \_M\_pback, 3008
  - \_M\_pback\_cur\_save, 3008
  - \_M\_pback\_end\_save, 3008
  - \_M\_pback\_init, 3008
  - \_M\_reading, 3008
  - \_M\_set\_buffer, 2991
  - ~stdio\_filebuf, 2991
- close, 2991
- eback, 2992
- egptr, 2992
- epptr, 2992
- fd, 2992
- file, 2993
- gbump, 2993
- getloc, 2993
- gptr, 2993
- imbue, 2994
- in\_avail, 2994
- is\_open, 2994
- open, 2994–2996
- overflow, 2996
- pbackfail, 2997
- pbase, 2997
- pbump, 2997
- pptr, 2998
- pubimbue, 2998
- pubseekoff, 2998
- pubseekpos, 2999
- pubsetbuf, 2999
- pubsync, 2999
- sbumpc, 2999
- seekoff, 2999
- seekpos, 3000

- setbuf, 3000
- setg, 3000
- setp, 3001
- sgetc, 3001
- sgetn, 3001
- showmanyc, 3002
- snextc, 3002
- sputbackc, 3002
- sputc, 3003
- sputn, 3003
- stdio\_filebuf, 2990
- sungetc, 3004
- sync, 3004
- uflow, 3004
- underflow, 3004
- xsggetn, 3005
- xspn, 3005
- \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3009
  - \_M\_buf\_locale, 3024
  - \_M\_in\_beg, 3024
  - \_M\_in\_cur, 3025
  - \_M\_in\_end, 3025
  - \_M\_out\_beg, 3025
  - \_M\_out\_cur, 3025
  - \_M\_out\_end, 3025
- eback, 3011
- egptr, 3011
- eptr, 3012
- file, 3012
- gbump, 3012
- getloc, 3012
- gptr, 3013
- imbue, 3013
- in\_avail, 3013
- overflow, 3014
- pbackfail, 3014
- pbase, 3015
- pbump, 3015
- pptr, 3015
- pubimbue, 3015
- pubseekoff, 3016
- pubseekpos, 3016
- pubsetbuf, 3016
- pubsync, 3017
- sbumpc, 3017
- seekoff, 3017
- seekpos, 3017
- setbuf, 3018
- setg, 3018
- setp, 3018
- sgetc, 3019
- sgetn, 3019
- showmanyc, 3019
- snextc, 3020
- sputbackc, 3020
- sputc, 3020
- sputn, 3022
- sungetc, 3022
- sync, 3022
- uflow, 3023
- underflow, 3023
- xsggetn, 3023
- xspn, 3024
- \_\_gnu\_cxx::subtractive\_rng, 3042
  - argument\_type, 3043
  - operator(), 3043
  - result\_type, 3043
  - subtractive\_rng, 3043
- \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 3046
  - \_M\_requested\_size, 3048
  - ~temporary\_buffer, 3047
  - begin, 3048
  - end, 3048
  - size, 3048
  - temporary\_buffer, 3047
- \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond >, 3052
- \_\_gnu\_cxx::throw\_allocator\_limit< \_Tp >, 3053
- \_\_gnu\_cxx::throw\_allocator\_random< \_Tp >, 3055
- \_\_gnu\_cxx::throw\_value\_base< \_Cond >, 3057
- \_\_gnu\_cxx::throw\_value\_limit, 3058
- \_\_gnu\_cxx::throw\_value\_random, 3059
- \_\_gnu\_cxx::typelist, 405
  - apply\_generator, 405
- \_\_gnu\_cxx::unary\_compose< \_Operation1, \_Operation2 >, 3136
  - argument\_type, 3137
  - result\_type, 3137
- \_\_gnu\_debug, 405
  - \_Distance\_precision, 412
  - \_base, 412
  - \_check\_singular, 412
  - \_check\_singular\_aux, 412
  - \_check\_string, 412
  - \_foreign\_iterator\_aux2, 413
  - \_get\_distance, 413
  - \_valid\_range, 413
  - \_valid\_range\_aux, 414
  - u16string, 411
  - u32string, 411
- \_\_gnu\_debug:: \_After\_nth\_from< \_Iterator >, 858
- \_\_gnu\_debug:: \_BeforeBeginHelper< \_Sequence >, 863
- \_\_gnu\_debug:: \_Equal\_to< \_Type >, 874
- \_\_gnu\_debug:: \_Not\_equal\_to< \_Type >, 940
- \_\_gnu\_debug:: \_Safe\_container< \_SafeContainer, \_Alloc, \_SafeBase, \_IsCxx11AllocatorAware >, 956
- \_\_gnu\_debug:: \_Safe\_forward\_list< \_SafeSequence >, 958

- [\\_M\\_const\\_iterators](#), 959
- [\\_M\\_detach\\_all](#), 959
- [\\_M\\_detach\\_singular](#), 959
- [\\_M\\_get\\_mutex](#), 959
- [\\_M\\_invalidate\\_if](#), 959
- [\\_M\\_iterators](#), 959
- [\\_M\\_revalidate\\_singular](#), 959
- [\\_M\\_swap](#), 959
- [\\_M\\_transfer\\_from\\_if](#), 959
- [\\_M\\_version](#), 960
- [\\_\\_gnu\\_debug:: Safe\\_iterator< \\_Iterator, \\_Sequence, \\_Category >](#), 960
- [\\_M\\_attach](#), 963
- [\\_M\\_attach\\_single](#), 964
- [\\_M\\_attached\\_to](#), 964
- [\\_M\\_before\\_dereferenceable](#), 964
- [\\_M\\_can\\_compare](#), 964
- [\\_M\\_dereferenceable](#), 964
- [\\_M\\_detach](#), 965
- [\\_M\\_detach\\_single](#), 965
- [\\_M\\_get\\_mutex](#), 965
- [\\_M\\_incrementable](#), 965
- [\\_M\\_invalidate](#), 965
- [\\_M\\_is\\_before\\_begin](#), 965
- [\\_M\\_is\\_begin](#), 965
- [\\_M\\_is\\_beginnest](#), 966
- [\\_M\\_is\\_end](#), 966
- [\\_M\\_next](#), 968
- [\\_M\\_prior](#), 968
- [\\_M\\_reset](#), 966
- [\\_M\\_sequence](#), 969
- [\\_M\\_singular](#), 966
- [\\_M\\_unlink](#), 966
- [\\_M\\_value\\_initialized](#), 966
- [\\_M\\_version](#), 969
- [\\_S\\_constant](#), 967
- [\\_Safe\\_iterator](#), 962, 963
- [base](#), 967
- [operator \\_Iterator](#), 967
- [operator++](#), 967, 968
- [operator->](#), 968
- [operator=](#), 968
- [operator\\*](#), 967
- [\\_\\_gnu\\_debug:: Safe\\_iterator\\_base](#), 969
- [\\_M\\_attach](#), 972
- [\\_M\\_attach\\_single](#), 972
- [\\_M\\_attached\\_to](#), 972
- [\\_M\\_can\\_compare](#), 972
- [\\_M\\_detach](#), 972
- [\\_M\\_detach\\_single](#), 972
- [\\_M\\_get\\_mutex](#), 973
- [\\_M\\_invalidate](#), 973
- [\\_M\\_next](#), 973
- [\\_M\\_prior](#), 973
- [\\_M\\_reset](#), 973
- [\\_M\\_sequence](#), 974
- [\\_M\\_singular](#), 973
- [\\_M\\_unlink](#), 973
- [\\_M\\_version](#), 974
- [\\_Safe\\_iterator\\_base](#), 971
- [\\_\\_gnu\\_debug:: Safe\\_local\\_iterator< \\_Iterator, \\_UContainer >](#), 974
- [\\_M\\_attach](#), 978
- [\\_M\\_attach\\_single](#), 978, 979
- [\\_M\\_attached\\_to](#), 979
- [\\_M\\_can\\_compare](#), 979
- [\\_M\\_dereferenceable](#), 979
- [\\_M\\_detach](#), 979
- [\\_M\\_detach\\_single](#), 979
- [\\_M\\_get\\_mutex](#), 979
- [\\_M\\_in\\_same\\_bucket](#), 979
- [\\_M\\_incrementable](#), 980
- [\\_M\\_invalidate](#), 980
- [\\_M\\_is\\_begin](#), 980
- [\\_M\\_is\\_end](#), 980
- [\\_M\\_next](#), 983
- [\\_M\\_prior](#), 983
- [\\_M\\_reset](#), 980
- [\\_M\\_sequence](#), 983
- [\\_M\\_singular](#), 980
- [\\_M\\_unlink](#), 980
- [\\_M\\_value\\_initialized](#), 981
- [\\_M\\_version](#), 983
- [\\_S\\_constant](#), 981
- [\\_Safe\\_local\\_iterator](#), 977
- [base](#), 981
- [bucket](#), 981
- [operator \\_Iterator](#), 981
- [operator++](#), 982
- [operator->](#), 982
- [operator=](#), 982
- [operator\\*](#), 981
- [\\_\\_gnu\\_debug:: Safe\\_local\\_iterator\\_base](#), 983
- [\\_M\\_attach](#), 986
- [\\_M\\_attach\\_single](#), 986
- [\\_M\\_attached\\_to](#), 986
- [\\_M\\_can\\_compare](#), 986
- [\\_M\\_detach](#), 987
- [\\_M\\_detach\\_single](#), 987
- [\\_M\\_get\\_mutex](#), 987
- [\\_M\\_invalidate](#), 987
- [\\_M\\_next](#), 988
- [\\_M\\_prior](#), 988
- [\\_M\\_reset](#), 987
- [\\_M\\_sequence](#), 988
- [\\_M\\_singular](#), 987
- [\\_M\\_unlink](#), 987
- [\\_M\\_version](#), 988

- [\\_Safe\\_local\\_iterator\\_base](#), 985
- [\\_\\_gnu\\_debug::\\_Safe\\_node\\_sequence< \\_Sequence >](#), 988
  - [\\_M\\_const\\_iterators](#), 991
  - [\\_M\\_detach\\_all](#), 990
  - [\\_M\\_detach\\_singular](#), 990
  - [\\_M\\_get\\_mutex](#), 990
  - [\\_M\\_invalidate\\_if](#), 990
  - [\\_M\\_iterators](#), 991
  - [\\_M\\_revalidate\\_singular](#), 990
  - [\\_M\\_swap](#), 990
  - [\\_M\\_transfer\\_from\\_if](#), 990
  - [\\_M\\_version](#), 991
- [\\_\\_gnu\\_debug::\\_Safe\\_sequence< \\_Sequence >](#), 991
  - [\\_M\\_const\\_iterators](#), 994
  - [\\_M\\_detach\\_all](#), 993
  - [\\_M\\_detach\\_singular](#), 993
  - [\\_M\\_get\\_mutex](#), 993
  - [\\_M\\_invalidate\\_all](#), 993
  - [\\_M\\_invalidate\\_if](#), 993
  - [\\_M\\_iterators](#), 994
  - [\\_M\\_revalidate\\_singular](#), 993
  - [\\_M\\_swap](#), 993
  - [\\_M\\_transfer\\_from\\_if](#), 994
  - [\\_M\\_version](#), 994
- [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base](#), 994
  - [\\_M\\_const\\_iterators](#), 997
  - [\\_M\\_detach\\_all](#), 996
  - [\\_M\\_detach\\_singular](#), 996
  - [\\_M\\_get\\_mutex](#), 996
  - [\\_M\\_invalidate\\_all](#), 996
  - [\\_M\\_iterators](#), 997
  - [\\_M\\_revalidate\\_singular](#), 997
  - [\\_M\\_swap](#), 997
  - [\\_M\\_version](#), 997
  - [~\\_Safe\\_sequence\\_base](#), 996
- [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container< \\_Container >](#), 997
  - [\\_M\\_const\\_iterators](#), 1000
  - [\\_M\\_const\\_local\\_iterators](#), 1000
  - [\\_M\\_detach\\_all](#), 999
  - [\\_M\\_detach\\_singular](#), 999
  - [\\_M\\_get\\_mutex](#), 999
  - [\\_M\\_invalidate\\_all](#), 999
  - [\\_M\\_invalidate\\_if](#), 999
  - [\\_M\\_invalidate\\_local\\_if](#), 1000
  - [\\_M\\_iterators](#), 1000
  - [\\_M\\_local\\_iterators](#), 1001
  - [\\_M\\_revalidate\\_singular](#), 1000
  - [\\_M\\_swap](#), 1000
  - [\\_M\\_version](#), 1001
- [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container\\_base](#), 1001
  - [\\_M\\_const\\_iterators](#), 1004
  - [\\_M\\_const\\_local\\_iterators](#), 1004
  - [\\_M\\_detach\\_all](#), 1003
  - [\\_M\\_detach\\_singular](#), 1003
  - [\\_M\\_get\\_mutex](#), 1003
  - [\\_M\\_invalidate\\_all](#), 1004
  - [\\_M\\_iterators](#), 1004
  - [\\_M\\_local\\_iterators](#), 1004
  - [\\_M\\_revalidate\\_singular](#), 1004
  - [\\_M\\_swap](#), 1004
  - [\\_M\\_version](#), 1005
  - [~\\_Safe\\_unordered\\_container\\_base](#), 1003
- [\\_\\_gnu\\_debug::\\_Safe\\_vector< \\_SafeSequence, \\_BaseSequence >](#), 1005
- [\\_\\_gnu\\_debug::\\_Sequence\\_traits< \\_Sequence >](#), 1008
- [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#), 1473
  - [\\_\\_resize\\_and\\_overwrite](#), 1481
  - [append](#), 1481, 1482
  - [assign](#), 1483, 1484
  - [at](#), 1485–1487
  - [back](#), 1488
  - [capacity](#), 1488
  - [compare](#), 1488–1494
  - [copy](#), 1494
  - [data](#), 1495
  - [empty](#), 1495
  - [erase](#), 1495, 1496
  - [find](#), 1497–1501
  - [find\\_first\\_not\\_of](#), 1501–1505
  - [find\\_first\\_of](#), 1506–1510
  - [find\\_last\\_not\\_of](#), 1510–1514
  - [find\\_last\\_of](#), 1514–1518
  - [front](#), 1519
  - [get\\_allocator](#), 1519
  - [insert](#), 1519–1525
  - [length](#), 1526
  - [max\\_size](#), 1526
  - [npos](#), 1545
  - [operator+=](#), 1526
  - [operator\[\]](#), 1526, 1527
  - [replace](#), 1527–1536
  - [reserve](#), 1537, 1538
  - [resize](#), 1539
  - [rfind](#), 1539–1543
  - [size](#), 1544
  - [substr](#), 1544
  - [swap](#), 1544
- [\\_\\_gnu\\_debug::type\\_info](#), 3131
  - [~type\\_info](#), 3132
  - [before](#), 3132
  - [name](#), 3132
- [\\_\\_gnu\\_internal](#), 414
- [\\_\\_gnu\\_parallel](#), 414
  - [\\_AlgorithmStrategy](#), 423
  - [\\_BinIndex](#), 422

- `__CASable`, 422
- `__CASable_bits`, 462
- `__CASable_mask`, 462
- `__FindAlgorithm`, 423
- `__MultiwayMergeAlgorithm`, 423
- `__Parallelism`, 423
- `__PartialSumAlgorithm`, 423
- `__SequenceIndex`, 422
- `__SortAlgorithm`, 423
- `__SplittingAlgorithm`, 423
- `__ThreadIndex`, 422
- `__calc_borders`, 423
- `__compare_and_swap`, 424
- `__decode2`, 424
- `__determine_samples`, 424
- `__encode2`, 425
- `__equally_split`, 425
- `__equally_split_point`, 426
- `__fetch_and_add`, 426
- `__find_template`, 426–428
- `__for_each_template_random_access`, 429
- `__for_each_template_random_access_ed`, 430
- `__for_each_template_random_access_omp_loop`, 430
- `__for_each_template_random_access_omp_loop_static`, 431
- `__for_each_template_random_access_workstealing`, 431
- `__is_sorted`, 432
- `__median_of_three_iterators`, 432
- `__merge_advance`, 433
- `__merge_advance_movc`, 433
- `__merge_advance_usual`, 434
- `__parallel_merge_advance`, 435
- `__parallel_nth_element`, 436
- `__parallel_partial_sort`, 436
- `__parallel_partial_sum`, 437
- `__parallel_partial_sum_basecase`, 437
- `__parallel_partial_sum_linear`, 437
- `__parallel_partition`, 438
- `__parallel_random_shuffle`, 438
- `__parallel_random_shuffle_drs`, 439
- `__parallel_random_shuffle_drs_pu`, 439
- `__parallel_sort`, 440–444
- `__parallel_sort_qs`, 445
- `__parallel_sort_qs_conquer`, 445
- `__parallel_sort_qs_divide`, 445
- `__parallel_sort_qsb`, 446
- `__parallel_unique_copy`, 446, 447
- `__qsb_conquer`, 447
- `__qsb_divide`, 448
- `__qsb_local_sort_with_helping`, 448
- `__random_number_pow2`, 449
- `__rd_log2`, 449
- `__round_up_to_pow2`, 449
- `__search_template`, 449
- `__sequential_multiway_merge`, 450
- `__sequential_random_shuffle`, 450
- `__shrink`, 451
- `__shrink_and_double`, 451
- `__yield`, 452
- `list_partition`, 452
- `max`, 452
- `min`, 452
- `multiseq_partition`, 453
- `multiseq_selection`, 453
- `multiway_merge`, 454
- `multiway_merge_3_variant`, 455
- `multiway_merge_4_variant`, 456
- `multiway_merge_exact_splitting`, 456
- `multiway_merge_loser_tree`, 457
- `multiway_merge_loser_tree_sentinel`, 457
- `multiway_merge_loser_tree_unguarded`, 458
- `multiway_merge_sampling_splitting`, 458
- `multiway_merge_sentinels`, 459
- `parallel_balanced`, 423
- `parallel_multiway_merge`, 460
- `parallel_omp_loop`, 423
- `parallel_omp_loop_static`, 423
- `parallel_sort_mwms`, 461
- `parallel_sort_mwms_pu`, 461
- `parallel_taskqueue`, 423
- `parallel_unbalanced`, 423
- `sequential`, 423
- `__gnu_parallel::_DRSSorterPU< __RAIter, __RandomNumberGenerator >`, 873
  - `_M_bins_begin`, 874
  - `_M_num_threads`, 874
  - `_M_sd`, 874
  - `_M_seed`, 874
  - `_bins_end`, 873
- `__gnu_parallel::_DRandomShufflingGlobalData< __RAIter >`, 871
  - `_DRandomShufflingGlobalData`, 872
  - `_M_bin_proc`, 872
  - `_M_dist`, 872
  - `_M_num_bins`, 872
  - `_M_num_bits`, 872
  - `_M_source`, 873
  - `_M_starts`, 873
  - `_M_temporaries`, 873
- `__gnu_parallel::_DummyReduct`, 874
- `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >`, 875
  - `first_argument_type`, 876
  - `result_type`, 876
  - `second_argument_type`, 876
- `__gnu_parallel::_EqualTo< _T1, _T2 >`, 876



[first\\_argument\\_type, 877](#)  
[result\\_type, 877](#)  
[second\\_argument\\_type, 877](#)  
[\\_\\_gnu\\_parallel::\\_\\_GuardedIterator< \\_RAIter, \\_Compare >, 886](#)  
[\\_\\_GuardedIterator, 886](#)  
[operator \\_RAIter, 887](#)  
[operator<, 887](#)  
[operator<=, 887](#)  
[operator++, 887](#)  
[operator\\*, 887](#)  
[\\_\\_gnu\\_parallel::\\_\\_IteratorPair< \\_Iterator1, \\_Iterator2, \\_IteratorCategory >, 893](#)  
[first, 899](#)  
[first\\_type, 895, 896](#)  
[operator<=, 896, 897](#)  
[operator==, 897, 898](#)  
[second, 899](#)  
[second\\_type, 896](#)  
[swap, 896, 898](#)  
[\\_\\_gnu\\_parallel::\\_\\_IteratorTriple< \\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory >, 899](#)  
[\\_\\_gnu\\_parallel::\\_\\_Job< \\_DifferenceTp >, 900](#)  
[\\_\\_M\\_first, 901](#)  
[\\_\\_M\\_last, 901](#)  
[\\_\\_M\\_load, 901](#)  
[\\_\\_gnu\\_parallel::\\_\\_Less< \\_T1, \\_T2 >, 902](#)  
[first\\_argument\\_type, 903](#)  
[result\\_type, 903](#)  
[second\\_argument\\_type, 903](#)  
[\\_\\_gnu\\_parallel::\\_\\_Lexicographic< \\_T1, \\_T2, \\_Compare >, 903](#)  
[first\\_argument\\_type, 904](#)  
[result\\_type, 904](#)  
[second\\_argument\\_type, 904](#)  
[\\_\\_gnu\\_parallel::\\_\\_LexicographicReverse< \\_T1, \\_T2, \\_Compare >, 905](#)  
[first\\_argument\\_type, 905](#)  
[result\\_type, 905](#)  
[second\\_argument\\_type, 906](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTree< \\_\\_stable, \\_Tp, \\_Compare >, 913](#)  
[\\_\\_M\\_log\\_k, 915](#)  
[\\_\\_delete\\_min\\_insert, 914](#)  
[\\_\\_get\\_min\\_source, 914](#)  
[\\_\\_insert\\_start, 914](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTree< false, \\_Tp, \\_Compare >, 915](#)  
[\\_\\_M\\_log\\_k, 917](#)  
[\\_\\_delete\\_min\\_insert, 916](#)  
[\\_\\_get\\_min\\_source, 916](#)  
[\\_\\_init\\_winner, 916](#)  
[\\_\\_insert\\_start, 917](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreeBase< \\_Tp, \\_Compare >, 917](#)  
[\\_\\_LoserTreeBase, 918](#)  
[\\_\\_M\\_comp, 919](#)  
[\\_\\_M\\_first\\_insert, 919](#)  
[\\_\\_M\\_log\\_k, 919](#)  
[\\_\\_M\\_losers, 919](#)  
[\\_\\_get\\_min\\_source, 919](#)  
[\\_\\_insert\\_start, 919](#)  
[~\\_LoserTreeBase, 918](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreeBase< \\_Tp, \\_Compare >::\\_\\_Loser, 912](#)  
[\\_\\_M\\_key, 912](#)  
[\\_\\_M\\_source, 912](#)  
[\\_\\_M\\_sup, 912](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreePointer< \\_\\_stable, \\_Tp, \\_Compare >, 920](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreePointer< false, \\_Tp, \\_Compare >, 921](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreePointerBase< \\_Tp, \\_Compare >, 922](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreePointerBase< \\_Tp, \\_Compare >::\\_\\_Loser, 913](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreePointerUnguarded< \\_\\_stable, \\_Tp, \\_Compare >, 922](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >, 923](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreePointerUnguardedBase< \\_Tp, \\_Compare >, 925](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreeTraits< \\_Tp >, 925](#)  
[\\_\\_M\\_use\\_pointer, 926](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreeUnguarded< \\_\\_stable, \\_Tp, \\_Compare >, 926](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >, 927](#)  
[\\_\\_gnu\\_parallel::\\_\\_LoserTreeUnguardedBase< \\_Tp, \\_Compare >, 928](#)  
[\\_\\_gnu\\_parallel::\\_\\_Multiplies< \\_Tp1, \\_Tp2, \\_Result >, 930](#)  
[first\\_argument\\_type, 931](#)  
[result\\_type, 931](#)  
[second\\_argument\\_type, 931](#)  
[\\_\\_gnu\\_parallel::\\_\\_Nothing, 941](#)  
[operator\(\), 941](#)  
[\\_\\_gnu\\_parallel::\\_\\_PMWMSortingData< \\_RAIter >, 944](#)  
[\\_\\_M\\_num\\_threads, 944](#)  
[\\_\\_M\\_offsets, 944](#)  
[\\_\\_M\\_pieces, 944](#)  
[\\_\\_M\\_samples, 944](#)  
[\\_\\_M\\_source, 945](#)  
[\\_\\_M\\_starts, 945](#)  
[\\_\\_M\\_temporary, 945](#)  
[\\_\\_gnu\\_parallel::\\_\\_Piece< \\_DifferenceTp >, 941](#)  
[\\_\\_M\\_begin, 942](#)  
[\\_\\_M\\_end, 942](#)  
[\\_\\_gnu\\_parallel::\\_\\_Plus< \\_Tp1, \\_Tp2, \\_Result >, 942](#)  
[first\\_argument\\_type, 943](#)

- result\_type, 943
- second\_argument\_type, 943
- \_\_gnu\_parallel::\_PseudoSequence< \_Tp, \_DifferenceTp >, 947
- \_PseudoSequence, 948
- begin, 948
- end, 948
- \_\_gnu\_parallel::\_PseudoSequenceIterator< \_Tp, \_DifferenceTp >, 948
- \_\_gnu\_parallel::\_QSBThreadLocal< \_RAIter >, 949
- \_M\_elements\_leftover, 950
- \_M\_global, 950
- \_M\_initial, 950
- \_M\_leftover\_parts, 950
- \_M\_num\_threads, 950
- \_Piece, 950
- \_QSBThreadLocal, 950
- \_\_gnu\_parallel::\_RandomNumber, 951
- \_RandomNumber, 951
- \_genrand\_bits, 952
- operator(), 952
- \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue< \_Tp >, 954
- \_RestrictedBoundedConcurrentQueue, 955
- ~\_RestrictedBoundedConcurrentQueue, 955
- pop\_back, 955
- pop\_front, 955
- push\_front, 956
- \_\_gnu\_parallel::\_SamplingSorter< \_\_stable, \_RAIter, \_StrictWeakOrdering >, 1006
- \_\_gnu\_parallel::\_SamplingSorter< false, \_RAIter, \_StrictWeakOrdering >, 1006
- \_\_gnu\_parallel::\_Settings, 1008
- accumulate\_minimal\_n, 1010
- adjacent\_difference\_minimal\_n, 1010
- cache\_line\_size, 1010
- count\_minimal\_n, 1011
- fill\_minimal\_n, 1011
- find\_increasing\_factor, 1011
- find\_initial\_block\_size, 1011
- find\_maximum\_block\_size, 1011
- find\_scale\_factor, 1011
- find\_sequential\_search\_size, 1011
- for\_each\_minimal\_n, 1011
- generate\_minimal\_n, 1011
- get, 1010
- L1\_cache\_size, 1012
- L2\_cache\_size, 1012
- max\_element\_minimal\_n, 1012
- merge\_minimal\_n, 1012
- merge\_oversampling, 1012
- min\_element\_minimal\_n, 1012
- multiway\_merge\_minimal\_k, 1012
- multiway\_merge\_minimal\_n, 1012
- multiway\_merge\_oversampling, 1012
- nth\_element\_minimal\_n, 1013
- partial\_sort\_minimal\_n, 1013
- partial\_sum\_dilation, 1013
- partial\_sum\_minimal\_n, 1013
- partition\_chunk\_share, 1013
- partition\_chunk\_size, 1013
- partition\_minimal\_n, 1013
- qsb\_steals, 1013
- random\_shuffle\_minimal\_n, 1013
- replace\_minimal\_n, 1014
- search\_minimal\_n, 1014
- set, 1010
- set\_difference\_minimal\_n, 1014
- set\_intersection\_minimal\_n, 1014
- set\_symmetric\_difference\_minimal\_n, 1014
- set\_union\_minimal\_n, 1014
- sort\_minimal\_n, 1014
- sort\_mwms\_oversampling, 1014
- sort\_qs\_num\_samples\_preset, 1014
- sort\_qsb\_base\_case\_maximal\_n, 1015
- TLB\_size, 1015
- transform\_minimal\_n, 1015
- unique\_copy\_minimal\_n, 1015
- \_\_gnu\_parallel::\_SplitConsistently< \_\_exact, \_RAIter, \_Compare, \_SortingPlacesIterator >, 1016
- \_\_gnu\_parallel::\_SplitConsistently< false, \_RAIter, \_Compare, \_SortingPlacesIterator >, 1016
- \_\_gnu\_parallel::\_SplitConsistently< true, \_RAIter, \_Compare, \_SortingPlacesIterator >, 1017
- \_\_gnu\_parallel::\_\_accumulate\_binop\_reduct< \_BinOp >, 728
- \_\_gnu\_parallel::\_\_accumulate\_selector< \_It >, 729
- \_M\_finish\_iterator, 730
- operator(), 730
- \_\_gnu\_parallel::\_\_adjacent\_difference\_selector< \_It >, 730
- \_M\_finish\_iterator, 731
- \_\_gnu\_parallel::\_\_adjacent\_find\_selector, 731
- \_M\_sequential\_algorithm, 732
- operator(), 732
- \_\_gnu\_parallel::\_\_binder1st< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 740
- argument\_type, 741
- result\_type, 741
- \_\_gnu\_parallel::\_\_binder2nd< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 741
- argument\_type, 742
- result\_type, 742
- \_\_gnu\_parallel::\_\_count\_if\_selector< \_It, \_Diff >, 746
- \_M\_finish\_iterator, 747
- operator(), 747

\_\_gnu\_parallel::\_\_count\_selector< \_It, \_Diff >, 747  
     \_M\_finish\_iterator, 749  
     operator(), 748  
 \_\_gnu\_parallel::\_\_fill\_selector< \_It >, 764  
     \_M\_finish\_iterator, 765  
     operator(), 765  
 \_\_gnu\_parallel::\_\_find\_first\_of\_selector< \_FIterator >, 765  
     \_M\_sequential\_algorithm, 766  
     operator(), 766  
 \_\_gnu\_parallel::\_\_find\_if\_selector, 767  
     \_M\_sequential\_algorithm, 767  
     operator(), 768  
 \_\_gnu\_parallel::\_\_for\_each\_selector< \_It >, 768  
     \_M\_finish\_iterator, 769  
     operator(), 769  
 \_\_gnu\_parallel::\_\_generate\_selector< \_It >, 770  
     \_M\_finish\_iterator, 771  
     operator(), 770  
 \_\_gnu\_parallel::\_\_generic\_find\_selector, 771  
 \_\_gnu\_parallel::\_\_generic\_for\_each\_selector< \_It >, 771  
     \_M\_finish\_iterator, 773  
 \_\_gnu\_parallel::\_\_identity\_selector< \_It >, 773  
     \_M\_finish\_iterator, 774  
     operator(), 774  
 \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp >, 774  
     \_M\_finish\_iterator, 776  
     \_\_begin1\_iterator, 776  
     \_\_begin2\_iterator, 776  
     \_\_inner\_product\_selector, 775  
     operator(), 775  
 \_\_gnu\_parallel::\_\_max\_element\_reduct< \_Compare, \_It >, 777  
 \_\_gnu\_parallel::\_\_min\_element\_reduct< \_Compare, \_It >, 777  
 \_\_gnu\_parallel::\_\_mismatch\_selector, 778  
     \_M\_sequential\_algorithm, 779  
     operator(), 779  
 \_\_gnu\_parallel::\_\_multiway\_merge\_3\_variant\_sentinel\_switch< \_\_sentinels, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 782  
 \_\_gnu\_parallel::\_\_multiway\_merge\_3\_variant\_sentinel\_switch< true, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 782  
 \_\_gnu\_parallel::\_\_multiway\_merge\_4\_variant\_sentinel\_switch< \_\_sentinels, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 783  
 \_\_gnu\_parallel::\_\_multiway\_merge\_4\_variant\_sentinel\_switch< true, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 783  
 \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel\_switch< \_\_sentinels, \_\_stable, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 783  
 \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel\_switch< false, \_\_stable, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 784  
 \_\_gnu\_parallel::\_\_replace\_if\_selector< \_It, \_Op, \_Tp >, 800  
     \_M\_finish\_iterator, 801  
     \_\_new\_val, 801  
     \_\_replace\_if\_selector, 801  
     operator(), 801  
 \_\_gnu\_parallel::\_\_replace\_selector< \_It, \_Tp >, 801  
     \_M\_finish\_iterator, 803  
     \_\_new\_val, 803  
     \_\_replace\_selector, 802  
     operator(), 803  
 \_\_gnu\_parallel::\_\_transform1\_selector< \_It >, 804  
     \_M\_finish\_iterator, 805  
     operator(), 804  
 \_\_gnu\_parallel::\_\_transform2\_selector< \_It >, 805  
     \_M\_finish\_iterator, 806  
     operator(), 805  
 \_\_gnu\_parallel::\_\_unary\_negate< \_Predicate, argument\_type >, 806  
     argument\_type, 807  
     result\_type, 807  
 \_\_gnu\_parallel::balanced\_quicksort\_tag, 1085  
     \_\_get\_num\_threads, 1086  
     set\_num\_threads, 1086  
 \_\_gnu\_parallel::balanced\_tag, 1087  
     \_\_get\_num\_threads, 1087  
     set\_num\_threads, 1087  
 \_\_gnu\_parallel::constant\_size\_blocks\_tag, 1886  
 \_\_gnu\_parallel::default\_parallel\_tag, 2023  
     \_\_get\_num\_threads, 2023  
     set\_num\_threads, 2023  
 \_\_gnu\_parallel::equal\_split\_tag, 2107  
 \_\_gnu\_parallel::exact\_tag, 2116  
     \_\_get\_num\_threads, 2117  
     set\_num\_threads, 2117  
 \_\_gnu\_parallel::find\_tag, 2132  
 \_\_gnu\_parallel::growing\_blocks\_tag, 2202  
 \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, 2523  
     \_\_get\_num\_threads, 2523  
     set\_num\_threads, 2523  
 \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, 2524  
     \_\_get\_num\_threads, 2524  
     set\_num\_threads, 2524  
 \_\_gnu\_parallel::multiway\_mergesort\_tag, 2525  
     \_\_get\_num\_threads, 2525  
     set\_num\_threads, 2525  
 \_\_gnu\_parallel::omp\_loop\_static\_tag, 2681  
     \_\_get\_num\_threads, 2682  
     set\_num\_threads, 2682  
 \_\_gnu\_parallel::omp\_loop\_tag, 2682  
     \_\_get\_num\_threads, 2683

- set\_num\_threads, 2683
- \_\_gnu\_parallel::parallel\_tag, 2717
  - \_\_get\_num\_threads, 2719
  - parallel\_tag, 2719
  - set\_num\_threads, 2719
- \_\_gnu\_parallel::quicksort\_tag, 2778
  - \_\_get\_num\_threads, 2778
  - set\_num\_threads, 2778
- \_\_gnu\_parallel::sampling\_tag, 2857
  - \_\_get\_num\_threads, 2857
  - set\_num\_threads, 2858
- \_\_gnu\_parallel::sequential\_tag, 2925
- \_\_gnu\_parallel::unbalanced\_tag, 3140
  - \_\_get\_num\_threads, 3140
  - set\_num\_threads, 3140
- \_\_gnu\_pbds, 462
- \_\_gnu\_pbds::associative\_tag, 1068
- \_\_gnu\_pbds::basic\_branch< Key, Mapped, Tag, Node\_Update, Policy\_Tl, \_Alloc >, 1088
- \_\_gnu\_pbds::basic\_branch\_tag, 1089
- \_\_gnu\_pbds::basic\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_Tl, \_Alloc >, 1160
- \_\_gnu\_pbds::basic\_hash\_tag, 1161
- \_\_gnu\_pbds::basic\_invalidatation\_guarantee, 1202
- \_\_gnu\_pbds::binary\_heap\_tag, 1744
- \_\_gnu\_pbds::binomial\_heap\_tag, 1757
- \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1776
  - cc\_hash\_max\_collision\_check\_resize\_trigger, 1777
  - external\_load\_access, 1777
  - get\_load, 1777
  - is\_grow\_needed, 1777
  - is\_resize\_needed, 1777
  - notify\_cleared, 1778
  - notify\_erase\_search\_collision, 1778
  - notify\_erase\_search\_end, 1778
  - notify\_erase\_search\_start, 1778
  - notify\_erased, 1778
  - notify\_externally\_resized, 1778
  - notify\_find\_search\_collision, 1778
  - notify\_find\_search\_end, 1779
  - notify\_find\_search\_start, 1779
  - notify\_insert\_search\_collision, 1779
  - notify\_insert\_search\_end, 1779
  - notify\_insert\_search\_start, 1779
  - notify\_inserted, 1779
  - notify\_resized, 1779
  - set\_load, 1780
- \_\_gnu\_pbds::cc\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >, 1780
  - cc\_hash\_table, 1781–1784
- \_\_gnu\_pbds::cc\_hash\_tag, 1784
- \_\_gnu\_pbds::container\_error, 1896
  - what, 1897
- \_\_gnu\_pbds::container\_tag, 1897
- \_\_gnu\_pbds::container\_traits< Cntnr >, 1898
  - erase\_can\_throw, 1899
  - order\_preserving, 1899
  - reverse\_iteration, 1899
  - split\_join\_can\_throw, 1899
- \_\_gnu\_pbds::container\_traits\_base< \_Tag >, 1899
- \_\_gnu\_pbds::container\_traits\_base< binary\_heap\_tag >, 1899
- \_\_gnu\_pbds::container\_traits\_base< binomial\_heap\_tag >, 1900
- \_\_gnu\_pbds::container\_traits\_base< cc\_hash\_tag >, 1900
- \_\_gnu\_pbds::container\_traits\_base< gp\_hash\_tag >, 1900
- \_\_gnu\_pbds::container\_traits\_base< list\_update\_tag >, 1901
- \_\_gnu\_pbds::container\_traits\_base< ov\_tree\_tag >, 1901
- \_\_gnu\_pbds::container\_traits\_base< pairing\_heap\_tag >, 1901
- \_\_gnu\_pbds::container\_traits\_base< pat\_trie\_tag >, 1901
- \_\_gnu\_pbds::container\_traits\_base< rb\_tree\_tag >, 1902
- \_\_gnu\_pbds::container\_traits\_base< rc\_binomial\_heap\_tag >, 1902
- \_\_gnu\_pbds::container\_traits\_base< splay\_tree\_tag >, 1902
- \_\_gnu\_pbds::container\_traits\_base< thin\_heap\_tag >, 1903
- \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_it< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >, 1720
- \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1722
  - const\_reference, 1723
  - difference\_type, 1723
  - get\_l\_child, 1724
  - get\_metadata, 1724
  - get\_r\_child, 1725
  - iterator\_category, 1724
  - metadata\_const\_reference, 1724
  - metadata\_type, 1724
  - operator!=, 1725
  - operator==, 1725
  - operator\*, 1725
  - reference, 1724
  - value\_type, 1724
- \_\_gnu\_pbds::detail::bin\_search\_tree\_it< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc

- >, [1725](#)
- \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1727](#)
- const\_reference, [1728](#)
- difference\_type, [1728](#)
- get\_l\_child, [1729](#)
- get\_metadata, [1729](#)
- get\_r\_child, [1729](#)
- iterator\_category, [1728](#)
- metadata\_const\_reference, [1728](#)
- metadata\_type, [1729](#)
- operator!=, [1729](#)
- operator==, [1730](#)
- operator\*, [1730](#)
- reference, [1729](#)
- value\_type, [1729](#)
- \_\_gnu\_pbds::detail::bin\_search\_tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, Node, \_Alloc >, [1730](#)
- node\_const\_iterator, [1731](#)
- \_\_gnu\_pbds::detail::bin\_search\_tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, Node, \_Alloc >, [1731](#)
- node\_const\_iterator, [1732](#)
- \_\_gnu\_pbds::detail::binary\_heap< Value\_Type, Cmp\_Fn, \_Alloc >, [1735](#)
- \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, [1737](#)
- binary\_heap\_const\_iterator\_, [1739](#)
- const\_pointer, [1738](#)
- const\_reference, [1738](#)
- difference\_type, [1738](#)
- iterator\_category, [1738](#)
- operator!=, [1739](#)
- operator->, [1740](#)
- operator==, [1740](#)
- operator\*, [1740](#)
- pointer, [1739](#)
- reference, [1739](#)
- value\_type, [1739](#)
- \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, [1740](#)
- binary\_heap\_point\_const\_iterator\_, [1743](#)
- const\_pointer, [1742](#)
- const\_reference, [1742](#)
- difference\_type, [1742](#)
- iterator\_category, [1742](#)
- operator!=, [1743](#)
- operator->, [1743](#)
- operator==, [1743](#)
- operator\*, [1743](#)
- pointer, [1742](#)
- reference, [1742](#)
- value\_type, [1742](#)
- \_\_gnu\_pbds::detail::binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc >, [1752](#)
- \_\_gnu\_pbds::detail::binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc >, [1754](#)
- \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Cltr, \_Alloc >, [1773](#)
- \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Itr, \_Alloc >, [1772](#)
- \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [1785](#)
- empty, [1788](#)
- get\_comb\_hash\_fn, [1788](#)
- get\_eq\_fn, [1789](#)
- get\_hash\_fn, [1789](#)
- get\_resize\_policy, [1789](#)
- \_\_gnu\_pbds::detail::cond\_dealtor< Entry, \_Alloc >, [1876](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, Tag, Policy\_TI >, [1887](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, cc\_hash\_tag, Policy\_TI >, [1890](#)
- type, [1890](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_TI >, [1890](#)
- type, [1891](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, list\_update\_tag, Policy\_TI >, [1891](#)
- type, [1891](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, ov\_tree\_tag, Policy\_TI >, [1891](#)
- type, [1892](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, pat\_trie\_tag, Policy\_TI >, [1892](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, rb\_tree\_tag, Policy\_TI >, [1892](#)
- type, [1892](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, splay\_tree\_tag, Policy\_TI >, [1893](#)
- type, [1893](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, cc\_hash\_tag, Policy\_TI >, [1893](#)
- type, [1893](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, gp\_hash\_tag, Policy\_TI >, [1894](#)
- type, [1894](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, list\_update\_tag, Policy\_TI >, [1894](#)
- type, [1894](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key,

null\_type, \_Alloc, ov\_tree\_tag, Policy\_Tl >, [1895](#)  
 type, [1895](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_Tl >, [1895](#)  
 type, [1895](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, rb\_tree\_tag, Policy\_Tl >, [1896](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_Tl >, [1896](#)  
 type, [1896](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type >, [1887](#)  
 type, [1888](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binomial\_heap\_tag, null\_type >, [1888](#)  
 type, [1888](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type >, [1888](#)  
 type, [1889](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type >, [1889](#)  
 type, [1889](#)  
 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type >, [1889](#)  
 type, [1890](#)  
 \_\_gnu\_pbds::detail::default\_comb\_hash\_fn, [2019](#)  
 type, [2019](#)  
 \_\_gnu\_pbds::detail::default\_eq\_fn< Key >, [2022](#)  
 type, [2022](#)  
 \_\_gnu\_pbds::detail::default\_hash\_fn< Key >, [2022](#)  
 type, [2022](#)  
 \_\_gnu\_pbds::detail::default\_probe\_fn< Comb\_Probe\_Fn >, [2024](#)  
 type, [2024](#)  
 \_\_gnu\_pbds::detail::default\_resize\_policy< Comb\_Hash\_Fn >, [2024](#)  
 type, [2024](#)  
 \_\_gnu\_pbds::detail::default\_trie\_access\_traits< Key >, [2025](#)  
 \_\_gnu\_pbds::detail::default\_trie\_access\_traits< std::basic\_string< Char, Char\_Traits, std::allocator< char > > >, [2025](#)  
 type, [2025](#)  
 \_\_gnu\_pbds::detail::default\_update\_policy, [2026](#)  
 type, [2026](#)  
 \_\_gnu\_pbds::detail::dumnode\_const\_iterator< Key, Data, \_Alloc >, [2067](#)  
 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, No\_Throw >, [2105](#)  
 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, false >, [2105](#)  
 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, false >::type, [3131](#)  
 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, true >, [2105](#)  
 type, [2106](#)  
 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, No\_Throw >, [2106](#)  
 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, false >, [2106](#)  
 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, true >, [2106](#)  
 \_\_gnu\_pbds::detail::eq\_by\_less< Key, Cmp\_Fn >, [2107](#)  
 \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [2192](#)  
 empty, [2194](#)  
 get\_comb\_probe\_fn, [2194](#), [2195](#)  
 get\_eq\_fn, [2195](#)  
 get\_hash\_fn, [2195](#)  
 get\_probe\_fn, [2195](#)  
 get\_resize\_policy, [2196](#)  
 \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, Store\_Hash >, [2221](#)  
 \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, false >, [2222](#)  
 \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, true >, [2222](#)  
 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, Hold\_Size >, [2226](#)  
 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, true >, [2226](#)  
 \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc >, [2302](#)  
 \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, [2304](#)  
 const\_pointer, [2305](#)  
 const\_reference, [2305](#)  
 difference\_type, [2305](#)  
 iterator\_category, [2306](#)  
 left\_child\_next\_sibling\_heap\_const\_iterator\_, [2306](#)  
 operator!=, [2306](#), [2307](#)  
 operator->, [2307](#)  
 operator==, [2307](#)  
 operator\*, [2307](#)  
 pointer, [2306](#)  
 reference, [2306](#)  
 value\_type, [2306](#)  
 \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node<



- [\\_Value, \\_Metadata, \\_Alloc >, 2307](#)
- [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator, Node, \\_Alloc >, 2308](#)
- [const\\_pointer, 2309](#)
- [const\\_reference, 2309](#)
- [difference\\_type, 2309](#)
- [iterator\\_category, 2310](#)
- [left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator, 2310](#)
- [operator!=, 2311](#)
- [operator->, 2311](#)
- [operator==, 2311](#)
- [operator\\*, 2311](#)
- [pointer, 2310](#)
- [reference, 2310](#)
- [value\\_type, 2310](#)
- [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_metadata< Size\\_Type >, 2368](#)
- [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type >, 2370](#)
- [\\_\\_gnu\\_pbds::detail::lu\\_map< Key, Mapped, Eq\\_Fn, \\_Alloc, Update\\_Policy >, 2371](#)
- [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >, 2407](#)
- [\\_\\_gnu\\_pbds::detail::maybe\\_null\\_type< Key, Mapped, \\_Alloc, Store\\_Hash >, 2416](#)
- [\\_\\_gnu\\_pbds::detail::maybe\\_null\\_type< Key, null\\_type, \\_Alloc, Store\\_Hash >, 2418](#)
- [\\_\\_gnu\\_pbds::detail::mod\\_based\\_range\\_hashing< Size\\_Type >, 2436](#)
- [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, Mapped >, 2534](#)
- [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, null\\_type >, 2535](#)
- [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >, 2694](#)
- [node\\_begin, 2696](#)
- [node\\_end, 2696](#)
- [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >::cond\\_dtor< Size\\_Type >, 1876](#)
- [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_const\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >, 2697](#)
- [get\\_l\\_child, 2698](#)
- [get\\_r\\_child, 2698](#)
- [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >, 2699](#)
- [get\\_l\\_child, 2700](#)
- [get\\_r\\_child, 2700](#)
- [operator\\*, 2700](#)
- [\\_\\_gnu\\_pbds::detail::pairing\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >, 2714](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base, 2730](#)
- [node\\_type, 2731](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Clter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >, 865](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Head< \\_ATraits, Metadata >, 888](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >, 889](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::const\\_iterator, 1879](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::iterator, 2296](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Iter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >, 891](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Leaf< \\_ATraits, Metadata >, 901](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< Metadata, \\_Alloc >, 929](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< null\\_type, \\_Alloc >, 929](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_base< \\_ATraits, Metadata >, 933](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >, 934](#)
- [get\\_child, 936](#)
- [get\\_metadata, 936](#)
- [metadata\\_const\\_reference, 935](#)
- [metadata\\_type, 935](#)
- [num\\_children, 936](#)
- [operator!=, 936](#)
- [operator==, 936](#)
- [operator\\*, 936](#)
- [valid\\_prefix, 937](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >, 937](#)
- [get\\_child, 938](#)
- [get\\_metadata, 939](#)
- [metadata\\_const\\_reference, 938](#)
- [metadata\\_type, 938](#)
- [num\\_children, 939](#)
- [operator!=, 939](#)
- [operator==, 939](#)
- [operator\\*, 939](#)
- [valid\\_prefix, 939](#)
- [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map< Key, Mapped, Node\\_And\\_It\\_Traits, \\_Alloc >, 2731](#)
- [node\\_begin, 2733](#)
- [node\\_end, 2734](#)
- [node\\_type, 2733](#)
- [\\_\\_gnu\\_pbds::detail::probe\\_fn\\_base< \\_Alloc >, 2769](#)
- [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, Store\\_Hash >, 2783](#)
- [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, false >, 2783](#)

- `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >`, [2784](#)
- `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >`, [2784](#)
- `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >`, [2785](#)
- `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >`, [2786](#)
- `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >`, [2786](#)
- `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >`, [2787](#)
- `__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >`, [2788](#)
- `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`, [2800](#)
  - `node_begin`, [2803](#)
  - `node_end`, [2803](#)
- `__gnu_pbds::detail::rb_tree_node< Value_Type, Metadata, _Alloc >`, [2803](#)
- `__gnu_pbds::detail::rc< _Node, _Alloc >`, [2805](#)
- `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`, [2806](#)
- `__gnu_pbds::detail::rebind_traits< _Alloc, T >`, [2808](#)
- `__gnu_pbds::detail::resize_policy< _Tp >`, [2832](#)
- `__gnu_pbds::detail::select_value_type< Key, Mapped >`, [2862](#)
- `__gnu_pbds::detail::select_value_type< Key, null_type >`, [2862](#)
- `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`, [2979](#)
  - `node_begin`, [2982](#)
  - `node_end`, [2982](#)
- `__gnu_pbds::detail::splay_tree_node< Value_Type, Metadata, _Alloc >`, [2982](#)
- `__gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >`, [3026](#)
- `__gnu_pbds::detail::stored_data< _Tv, _Th, false >`, [3027](#)
- `__gnu_pbds::detail::stored_hash< _Th >`, [3027](#)
- `__gnu_pbds::detail::stored_value< _Tv >`, [3028](#)
- `__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >`, [3043](#)
- `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`, [3048](#)
- `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >`, [3095](#)
- `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >`, [3095](#)
- `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >`, [3095](#)
- `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`, [3096](#)
- `__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >`, [3098](#)
- `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`, [3099](#)
  - `node_const_iterator`, [3099](#)
- `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`, [3099](#)
  - `node_const_iterator`, [3100](#)
- `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`, [3100](#)
  - `node_const_iterator`, [3102](#)
- `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`, [3102](#)
  - `node_const_iterator`, [3102](#)
- `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`, [3102](#)
  - `node_const_iterator`, [3104](#)
- `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`, [3104](#)
  - `node_const_iterator`, [3105](#)
- `__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >`, [3108](#)
- `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >`, [3108](#)
- `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`, [3108](#)
- `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`, [3109](#)
- `__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, [3112](#)
  - `end`, [3113](#)
- `__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >`, [3119](#)
- `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`, [3120](#)
  - `node_const_iterator`, [3120](#)
  - `node_update`, [3120](#)
  - `synth_access_traits`, [3120](#)
- `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >`, [3121](#)
  - `node_const_iterator`, [3121](#)
  - `node_update`, [3121](#)
  - `synth_access_traits`, [3122](#)
- `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`, [3133](#)
- `__gnu_pbds::direct_mask_range_hashing< Size_Type >`, [2051](#)
  - `operator()`, [2052](#)
- `__gnu_pbds::direct_mod_range_hashing< Size_Type >`, [2053](#)
  - `operator()`, [2053](#)
- `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn,`



- Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Re-size\_Policy, Store\_Hash, \_Alloc >, 2186
- gp\_hash\_table, 2187–2190
- \_\_gnu\_pbds::gp\_hash\_tag, 2191
- \_\_gnu\_pbds::hash\_exponential\_size\_policy< Size\_Type >, 2223
  - hash\_exponential\_size\_policy, 2223
- \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2224
  - external\_load\_access, 2225
  - get\_loads, 2225
  - hash\_load\_check\_resize\_trigger, 2225
  - notify\_cleared, 2225
  - notify\_inserted, 2225
  - notify\_resized, 2225
  - set\_loads, 2226
- \_\_gnu\_pbds::hash\_prime\_size\_policy, 2232
  - hash\_prime\_size\_policy, 2232
  - size\_type, 2232
- \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, 2234
  - external\_load\_access, 2235
  - get\_actual\_size, 2236
  - get\_loads, 2236
  - get\_new\_size, 2236
  - get\_size\_policy, 2237
  - get\_trigger\_policy, 2237
  - hash\_standard\_resize\_policy, 2236
  - resize, 2237
  - set\_loads, 2237
- \_\_gnu\_pbds::insert\_error, 2250
  - what, 2251
- \_\_gnu\_pbds::join\_error, 2301
  - what, 2302
- \_\_gnu\_pbds::linear\_probe\_fn< Size\_Type >, 2323
  - operator(), 2324
- \_\_gnu\_pbds::list\_update< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >, 2347
  - list\_update, 2348
- \_\_gnu\_pbds::list\_update\_tag, 2348
- \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, 2369
  - max\_count, 2370
  - metadata\_reference, 2369
  - metadata\_type, 2370
  - operator(), 2370
- \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, 2374
  - metadata\_reference, 2374
  - metadata\_type, 2374
  - operator(), 2374
- \_\_gnu\_pbds::null\_node\_update< \_Tp1, \_Tp2, \_Tp3, \_Tp4 >, 2541
- \_\_gnu\_pbds::null\_type, 2542
- \_\_gnu\_pbds::ov\_tree\_tag, 2700
- \_\_gnu\_pbds::pairing\_heap\_tag, 2717
- \_\_gnu\_pbds::pat\_trie\_tag, 2734
- \_\_gnu\_pbds::point\_invalidation\_guarantee, 2751
- \_\_gnu\_pbds::priority\_queue< \_Tv, Cmp\_Fn, Tag, \_Alloc >, 2763
  - priority\_queue, 2764
- \_\_gnu\_pbds::priority\_queue\_tag, 2768
- \_\_gnu\_pbds::quadratic\_probe\_fn< Size\_Type >, 2774
  - operator(), 2774
- \_\_gnu\_pbds::range\_invalidation\_guarantee, 2782
- \_\_gnu\_pbds::rb\_tree\_tag, 2804
- \_\_gnu\_pbds::rc\_binomial\_heap\_tag, 2808
- \_\_gnu\_pbds::resize\_error, 2831
  - what, 2831
- \_\_gnu\_pbds::sample\_probe\_fn, 2843
  - operator(), 2844
  - sample\_probe\_fn, 2844
  - swap, 2844
- \_\_gnu\_pbds::sample\_range\_hashing, 2844
  - notify\_resized, 2845
  - operator(), 2845
  - sample\_range\_hashing, 2845
  - size\_type, 2845
  - swap, 2845
- \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 2845
  - notify\_resized, 2846
  - operator(), 2846
  - sample\_ranged\_hash\_fn, 2846
  - swap, 2846
- \_\_gnu\_pbds::sample\_ranged\_probe\_fn, 2846
- \_\_gnu\_pbds::sample\_resize\_policy, 2847
  - get\_new\_size, 2848
  - is\_resize\_needed, 2848
  - notify\_cleared, 2848
  - notify\_erase\_search\_collision, 2848
  - notify\_erase\_search\_end, 2848
  - notify\_erase\_search\_start, 2848
  - notify\_erased, 2848
  - notify\_find\_search\_collision, 2848
  - notify\_find\_search\_end, 2849
  - notify\_find\_search\_start, 2849
  - notify\_insert\_search\_collision, 2849
  - notify\_insert\_search\_end, 2849
  - notify\_insert\_search\_start, 2849
  - notify\_inserted, 2849
  - notify\_resized, 2849
  - sample\_range\_hashing, 2849
  - sample\_resize\_policy, 2848
  - size\_type, 2848
  - swap, 2849
- \_\_gnu\_pbds::sample\_resize\_trigger, 2850
  - is\_grow\_needed, 2851
  - is\_resize\_needed, 2851

[notify\\_cleared](#), 2851  
[notify\\_erase\\_search\\_collision](#), 2851  
[notify\\_erase\\_search\\_end](#), 2851  
[notify\\_erase\\_search\\_start](#), 2851  
[notify\\_erased](#), 2851  
[notify\\_externally\\_resized](#), 2851  
[notify\\_find\\_search\\_collision](#), 2851  
[notify\\_find\\_search\\_end](#), 2851  
[notify\\_find\\_search\\_start](#), 2851  
[notify\\_insert\\_search\\_collision](#), 2852  
[notify\\_insert\\_search\\_end](#), 2852  
[notify\\_insert\\_search\\_start](#), 2852  
[notify\\_inserted](#), 2852  
[notify\\_resized](#), 2852  
[sample\\_range\\_hashing](#), 2852  
[sample\\_resize\\_trigger](#), 2850  
[size\\_type](#), 2850  
[swap](#), 2852  
[\\_\\_gnu\\_pbds::sample\\_size\\_policy](#), 2852  
[get\\_nearest\\_larger\\_size](#), 2853  
[get\\_nearest\\_smaller\\_size](#), 2853  
[sample\\_range\\_hashing](#), 2853  
[sample\\_size\\_policy](#), 2853  
[size\\_type](#), 2853  
[swap](#), 2853  
[\\_\\_gnu\\_pbds::sample\\_tree\\_node\\_update< Const\\_Node\\_Itr, Node\\_Itr, Cmp\\_Fn, \\_Alloc >](#), 2854  
[\\_\\_gnu\\_pbds::sample\\_trie\\_access\\_traits](#), 2854  
[begin](#), 2854  
[e\\_pos](#), 2854  
[e\\_type](#), 2854  
[end](#), 2855  
[\\_\\_gnu\\_pbds::sample\\_trie\\_node\\_update< Node\\_Cltr, Node\\_Itr, \\_ATraits, \\_Alloc >](#), 2855  
[operator\(\)](#), 2855  
[sample\\_trie\\_node\\_update](#), 2855  
[\\_\\_gnu\\_pbds::sample\\_update\\_policy](#), 2856  
[metadata\\_type](#), 2856  
[operator\(\)](#), 2856  
[sample\\_update\\_policy](#), 2856  
[swap](#), 2857  
[\\_\\_gnu\\_pbds::sequence\\_tag](#), 2924  
[\\_\\_gnu\\_pbds::splay\\_tree\\_tag](#), 2983  
[\\_\\_gnu\\_pbds::string\\_tag](#), 3029  
[\\_\\_gnu\\_pbds::thin\\_heap\\_tag](#), 3050  
[\\_\\_gnu\\_pbds::tree< Key, Mapped, Cmp\\_Fn, Tag, Node\\_Update, \\_Alloc >](#), 3092  
[cmp\\_fn](#), 3094  
[tree](#), 3094  
[\\_\\_gnu\\_pbds::tree\\_order\\_statistics\\_node\\_update< Node\\_Cltr, Node\\_Itr, Cmp\\_Fn, \\_Alloc >](#), 3096  
[find\\_by\\_order](#), 3097  
[operator\(\)](#), 3097  
[order\\_of\\_key](#), 3097  
[\\_\\_gnu\\_pbds::tree\\_tag](#), 3098  
[\\_\\_gnu\\_pbds::trie< Key, Mapped, \\_ATraits, Tag, Node\\_Update, \\_Alloc >](#), 3105  
[access\\_traits](#), 3107  
[trie](#), 3107  
[\\_\\_gnu\\_pbds::trie\\_order\\_statistics\\_node\\_update< Node\\_Cltr, Node\\_Itr, \\_ATraits, \\_Alloc >](#), 3109  
[find\\_by\\_order](#), 3111  
[operator\(\)](#), 3111  
[order\\_of\\_key](#), 3111  
[order\\_of\\_prefix](#), 3111  
[\\_\\_gnu\\_pbds::trie\\_prefix\\_search\\_node\\_update< Node\\_Cltr, Node\\_Itr, \\_ATraits, \\_Alloc >](#), 3113  
[a\\_const\\_iterator](#), 3115  
[access\\_traits](#), 3115  
[allocator\\_type](#), 3115  
[operator\(\)](#), 3115  
[prefix\\_range](#), 3115, 3116  
[size\\_type](#), 3115  
[\\_\\_gnu\\_pbds::trie\\_string\\_access\\_traits< String, Min\\_E\\_Val, Max\\_E\\_Val, Reverse, \\_Alloc >](#), 3116  
[begin](#), 3117  
[const\\_iterator](#), 3117  
[e\\_pos](#), 3118  
[e\\_type](#), 3117  
[end](#), 3118  
[\\_\\_gnu\\_pbds::trie\\_tag](#), 3118  
[\\_\\_gnu\\_pbds::trivial\\_iterator\\_tag](#), 3122  
[\\_\\_gnu\\_sequential](#), 464  
[\\_\\_init\\_winner](#)  
[\\_\\_gnu\\_parallel::\\_LoserTree< false, \\_Tp, \\_Compare >](#), 916  
[\\_\\_inner\\_product\\_selector](#)  
[\\_\\_gnu\\_parallel::\\_inner\\_product\\_selector< \\_It, \\_It2, \\_Tp >](#), 775  
[\\_\\_inplace\\_stable\\_sort](#)  
[std](#), 582  
[\\_\\_insert\\_start](#)  
[\\_\\_gnu\\_parallel::\\_LoserTree< \\_\\_stable, \\_Tp, \\_Compare >](#), 914  
[\\_\\_gnu\\_parallel::\\_LoserTree< false, \\_Tp, \\_Compare >](#), 917  
[\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >](#), 919  
[\\_\\_int\\_traits](#)  
[\\_\\_gnu\\_cxx](#), 394  
[\\_\\_invoke](#)  
[Utilities](#), 299  
[\\_\\_is\\_sorted](#)  
[\\_\\_gnu\\_parallel](#), 432  
[\\_\\_iterator\\_category](#)  
[Iterators](#), 170  
[\\_\\_lg](#)  
[std](#), 582

- \_\_median
  - SGI, [150](#)
- \_\_median\_of\_three\_iterators
  - \_\_gnu\_parallel, [432](#)
- \_\_merge\_adaptive
  - std, [582](#)
- \_\_merge\_advance
  - \_\_gnu\_parallel, [433](#)
- \_\_merge\_advance\_movc
  - \_\_gnu\_parallel, [433](#)
- \_\_merge\_advance\_usual
  - \_\_gnu\_parallel, [434](#)
- \_\_merge\_without\_buffer
  - std, [583](#)
- \_\_move\_median\_to\_first
  - std, [583](#)
- \_\_move\_merge
  - std, [583](#)
- \_\_move\_merge\_adaptive
  - std, [583](#)
- \_\_move\_merge\_adaptive\_backward
  - std, [584](#)
- \_\_multiline
  - std::regex\_constants, [715](#)
- \_\_new\_val
  - \_\_gnu\_parallel::\_\_replace\_if\_selector<\_It, \_Op, \_Tp>, [801](#)
  - \_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp>, [803](#)
- \_\_num\_bitmaps
  - \_\_gnu\_cxx::\_\_detail, [404](#)
- \_\_num\_blocks
  - \_\_gnu\_cxx::\_\_detail, [405](#)
- \_\_num\_get\_type
  - std::basic\_ios<\_CharT, \_Traits>, [1207](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [1354](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [1387](#)
  - std::basic\_ostream<\_CharT, \_Traits>::sentry, [2901](#)
  - std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc>, [1419](#)
- \_\_num\_put\_type
  - std::basic\_ifstream<\_CharT, \_Traits>, [1168](#)
  - std::basic\_ios<\_CharT, \_Traits>, [1207](#)
  - std::basic\_istream<\_CharT, \_Traits>, [1277](#)
  - std::basic\_istream<\_CharT, \_Traits>::sentry, [2868](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1314](#)
- \_\_parallel\_merge\_advance
  - \_\_gnu\_parallel, [435](#)
- \_\_parallel\_nth\_element
  - \_\_gnu\_parallel, [436](#)
- \_\_parallel\_partial\_sort
  - \_\_gnu\_parallel, [436](#)
- \_\_parallel\_partial\_sum
  - \_\_gnu\_parallel, [437](#)
- \_\_parallel\_partial\_sum\_basecase
  - \_\_gnu\_parallel, [437](#)
- \_\_parallel\_partial\_sum\_linear
  - \_\_gnu\_parallel, [437](#)
- \_\_parallel\_partition
  - \_\_gnu\_parallel, [438](#)
- \_\_parallel\_random\_shuffle
  - \_\_gnu\_parallel, [438](#)
- \_\_parallel\_random\_shuffle\_drs
  - \_\_gnu\_parallel, [439](#)
- \_\_parallel\_random\_shuffle\_drs\_pu
  - \_\_gnu\_parallel, [439](#)
- \_\_parallel\_sort
  - \_\_gnu\_parallel, [440–444](#)
- \_\_parallel\_sort\_qs
  - \_\_gnu\_parallel, [445](#)
- \_\_parallel\_sort\_qs\_conquer
  - \_\_gnu\_parallel, [445](#)
- \_\_parallel\_sort\_qs\_divide
  - \_\_gnu\_parallel, [445](#)
- \_\_parallel\_sort\_qsb
  - \_\_gnu\_parallel, [446](#)
- \_\_parallel\_unique\_copy
  - \_\_gnu\_parallel, [446](#), [447](#)
- \_\_partition
  - std, [584](#)
- \_\_polynomial
  - std::regex\_constants, [715](#)
- \_\_ptr\_rebind
  - std, [578](#)
- \_\_qsb\_conquer
  - \_\_gnu\_parallel, [447](#)
- \_\_qsb\_divide
  - \_\_gnu\_parallel, [448](#)
- \_\_qsb\_local\_sort\_with\_helping
  - \_\_gnu\_parallel, [448](#)
- \_\_random\_number\_pow2
  - \_\_gnu\_parallel, [449](#)
- \_\_rd\_log2
  - \_\_gnu\_parallel, [449](#)
- \_\_replace\_if\_selector
  - \_\_gnu\_parallel::\_\_replace\_if\_selector<\_It, \_Op, \_Tp>, [801](#)
- \_\_replace\_selector
  - \_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp>, [802](#)
- \_\_resize\_and\_overwrite
  - \_\_gnu\_debug::basic\_string<\_CharT, \_Traits, \_Allocator>, [1481](#)
  - std::basic\_string<\_CharT, \_Traits, \_Alloc>, [1561](#)
- \_\_reverse
  - std, [584](#)
- \_\_rotate
  - std, [585](#)
- \_\_rotate\_adaptive

- std, 585
- \_\_round\_up\_to\_pow2
  - \_\_gnu\_parallel, 449
- \_\_sample
  - std, 585, 586
- \_\_search\_n\_aux
  - std, 586
- \_\_search\_template
  - \_\_gnu\_parallel, 449
- \_\_sequential\_multiway\_merge
  - \_\_gnu\_parallel, 450
- \_\_sequential\_random\_shuffle
  - \_\_gnu\_parallel, 450
- \_\_shrink
  - \_\_gnu\_parallel, 451
- \_\_shrink\_and\_double
  - \_\_gnu\_parallel, 451
- \_\_stable\_partition\_adaptive
  - std, 586
- \_\_static\_pointer\_cast
  - \_\_gnu\_cxx, 394
- \_\_streambuf\_type
  - std::basic\_streambuf< \_CharT, \_Traits >, 1459
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3330
- \_\_umap\_traits
  - std, 578
- \_\_ummap\_traits
  - std, 579
- \_\_umset\_traits
  - std, 579
- \_\_uset\_traits
  - std, 579
- \_\_valid\_range
  - \_\_gnu\_debug, 413
- \_\_valid\_range\_aux
  - \_\_gnu\_debug, 414
- \_\_verbose\_terminate\_handler
  - Exceptions, 129
- \_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 811–813
- \_\_yield
  - \_\_gnu\_parallel, 452
- ~\_LoserTreeBase
  - \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >, 918
- ~\_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue< \_Tp >, 955
- ~\_Safe\_sequence\_base
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 996
- ~\_Safe\_unordered\_container\_base
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 1003
- ~\_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 814
- ~any
  - std::experimental::fundamentals\_v1::any, 1065
- ~auto\_ptr
  - std::auto\_ptr< \_Tp >, 1073
- ~basic\_filebuf
  - std::basic\_filebuf< \_CharT, \_Traits >, 1093
- ~basic\_fstream
  - std::basic\_fstream< \_CharT, \_Traits >, 1119
- ~basic\_ifstream
  - std::basic\_ifstream< \_CharT, \_Traits >, 1170
- ~basic\_ios
  - std::basic\_ios< \_CharT, \_Traits >, 1209
- ~basic\_iostream
  - std::basic\_iostream< \_CharT, \_Traits >, 1232
- ~basic\_istream
  - std::basic\_istream< \_CharT, \_Traits >, 1278
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2870
- ~basic\_istreamstringstream
  - std::basic\_istreamstringstream< \_CharT, \_Traits, \_Alloc >, 1317
- ~basic\_ofstream
  - std::basic\_ofstream< \_CharT, \_Traits >, 1356
- ~basic\_ostream
  - std::basic\_ostream< \_CharT, \_Traits >, 1388
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2903
- ~basic\_ostringstream
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1421
- ~basic\_regex
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1450
- ~basic\_streambuf
  - std::basic\_streambuf< \_CharT, \_Traits >, 1460
- ~basic\_string
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1557, 1560
- ~basic\_stringstream
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1677
- ~collate
  - std::\_\_cxx11::collate< \_CharT >, 1858
- ~ctype
  - std::ctype< char >, 1927
  - std::ctype< wchar\_t >, 1961
- ~deque
  - std::deque< \_Tp, \_Alloc >, 2036
- ~facet
  - std::locale::facet, 2128
- ~forward\_list
  - std::forward\_list< \_Tp, \_Alloc >, 2146
- ~gslice
  - Numeric Arrays, 221

- ~ios\_base
  - std::ios\_base, [2261](#)
- ~list
  - std::list< \_Tp, \_Alloc >, [2333](#)
- ~locale
  - std::locale, [2352](#)
- ~map
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2385](#)
- ~match\_results
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2411](#)
- ~messages
  - std::messages< \_CharT >, [2430](#)
- ~money\_get
  - std::money\_get< \_CharT, \_InIter >, [2441](#)
- ~money\_put
  - std::money\_put< \_CharT, \_OutIter >, [2446](#)
- ~moneypunct
  - std::moneypunct< \_CharT, \_Intl >, [2451](#)
- ~multimap
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [2481](#)
- ~multiset
  - std::multiset< \_Key, \_Compare, \_Alloc >, [2508](#)
- ~num\_get
  - std::num\_get< \_CharT, \_InIter >, [2546](#)
- ~num\_put
  - std::num\_put< \_CharT, \_OutIter >, [2562](#)
- ~numpunct
  - std::numpunct< \_CharT >, [2674](#)
- ~sentry
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2903](#)
- ~set
  - std::set< \_Key, \_Compare, \_Alloc >, [2936](#)
- ~stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [2991](#)
- ~temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, [3047](#)
- ~time\_get
  - std::time\_get< \_CharT, \_InIter >, [3063](#)
- ~time\_put
  - std::time\_put< \_CharT, \_OutIter >, [3085](#)
- ~type\_info
  - \_\_gnu\_debug::type\_info, [3132](#)
  - std::type\_info, [3133](#)
- ~unique\_ptr
  - std::unique\_ptr< \_Tp, \_Dp >, [3151](#)
  - std::unique\_ptr< \_Tp[], \_Dp >, [3157](#), [3159](#)
- ~vector
  - std::vector< \_Tp, \_Alloc >, [3294](#)
  - std::vector< bool, \_Alloc >, [3315](#)
- a
  - std::extreme\_value\_distribution< \_RealType >, [2125](#)
  - std::weibull\_distribution< \_RealType >, [3346](#)
- a\_const\_iterator
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, [3115](#)
- abi, [464](#)
- abs
  - Complex Numbers, [181](#)
  - math.h, [3700](#)
  - stdlib.h, [3756](#)
  - Time, [366](#)
- access\_traits
  - \_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >, [3107](#)
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, [3115](#)
- accumulate
  - Generalized Numeric operations, [6](#), [7](#)
- accumulate\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1010](#)
- acosh
  - std, [587](#)
- Adaptors for pointers to functions, [309](#)
  - ptr\_fun, [309](#)
- Adaptors for pointers to members, [310](#)
- add\_lvalue\_reference\_t
  - Metaprogramming, [350](#)
- add\_pointer\_t
  - Metaprogramming, [350](#)
- add\_rvalue\_reference\_t
  - Metaprogramming, [350](#)
- addressof
  - Utilities, [299](#)
- adjacent\_difference
  - Generalized Numeric operations, [7](#), [8](#)
- adjacent\_difference\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1010](#)
- adjacent\_find
  - Non-Mutating, [44](#)
- adjustfield
  - std::basic\_fstream< \_CharT, \_Traits >, [1155](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1198](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1221](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1267](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1304](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2893](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1344](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1378](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1408](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2921](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1442](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1712](#)

- std::ios\_base, [2266](#)
- adopt\_lock
  - Mutexes, [118](#)
- advance
  - std, [587](#)
- airy\_ai
  - Mathematical Special Functions, [192](#)
- airy\_aif
  - Mathematical Special Functions, [192](#)
- airy\_ail
  - Mathematical Special Functions, [192](#)
- airy\_bi
  - Mathematical Special Functions, [192](#)
- airy\_bif
  - Mathematical Special Functions, [192](#)
- airy\_bil
  - Mathematical Special Functions, [192](#)
- algo.h, [3713](#)
- algbase.h, [3722](#)
- algorithm, [3351–3353](#)
- algorithmfwd.h, [3363](#), [3368](#)
- Algorithms, [4](#)
- align
  - Memory, [318](#)
- align.h, [3376](#)
- aligned\_buffer.h, [3604](#)
- aligned\_storage\_t
  - Metaprogramming, [350](#)
- alignment\_value
  - Metaprogramming, [353](#)
- all
  - std::bitset< \_Nb >, [1765](#)
  - std::locale, [2356](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2074](#)
- all\_of
  - Non-Mutating, [44](#)
- alloc\_traits.h, [3376](#), [3377](#)
- allocate
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [735](#), [736](#)
  - std::allocator\_traits< \_Alloc >, [1034](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1041](#)
  - std::allocator\_traits< allocator< void > >, [1049](#), [1051](#)
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, [1058](#), [1059](#)
- allocate\_shared
  - Pointer Abstractions, [328](#)
- allocated\_ptr.h, [3377](#)
- allocator.h, [3377](#)
- allocator\_type
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, [3115](#)
  - std::allocator\_traits< \_Alloc >, [1032](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1038](#)
  - std::allocator\_traits< allocator< void > >, [1047](#)
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, [1055](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [2931](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3170](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3200](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3229](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3255](#)
- Allocators, [323](#)
  - \_\_allocator\_base, [324](#)
  - operator==, [324](#)
- alpha
  - std::gamma\_distribution< \_RealType >, [2181](#)
- any, [3353](#), [3354](#)
  - std::bitset< \_Nb >, [1765](#)
  - std::experimental::fundamentals\_v1::any, [1065](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2074](#)
- any\_cast
  - Type-safe container of any type, [288–290](#)
- any\_of
  - Non-Mutating, [45](#)
- app
  - std::basic\_fstream< \_CharT, \_Traits >, [1155](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1198](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1221](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1267](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1304](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2893](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1344](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1378](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1408](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2921](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1442](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1712](#)
  - std::ios\_base, [2266](#)
- append
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [814–816](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1481](#), [1482](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1561–1566](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2074](#), [2075](#)
- apply
  - Numeric Arrays, [221](#), [222](#)

- apply\_generator
  - \_\_gnu\_cxx::typelist, 405
- arg
  - Complex Numbers, 181
  - std::tr1, 724
- argument\_type
  - \_\_gnu\_cxx::binary\_compose< \_Operation1, \_Operation2, \_Operation3 >, 1733
  - \_\_gnu\_cxx::select1st< \_Pair >, 2861
  - \_\_gnu\_cxx::select2nd< \_Pair >, 2861
  - \_\_gnu\_cxx::subtractive\_rng, 3043
  - \_\_gnu\_cxx::unary\_compose< \_Operation1, \_Operation2 >, 3137
  - \_\_gnu\_parallel::\_\_binder1st< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 741
  - \_\_gnu\_parallel::\_\_binder2nd< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 742
  - \_\_gnu\_parallel::\_\_unary\_negate< \_Predicate, argument\_type >, 807
  - std::binder1st< \_Operation >, 1746
  - std::binder2nd< \_Operation >, 1748
  - std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >, 1884
  - std::const\_mem\_fun\_t< \_Ret, \_Tp >, 1885
  - std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 2209
  - std::hash< \_\_gnu\_cxx::throw\_value\_random >, 2210
  - std::logical\_not< \_Tp >, 2362
  - std::logical\_not< void >, 2363
  - std::mem\_fun\_ref\_t< \_Ret, \_Tp >, 2421
  - std::mem\_fun\_t< \_Ret, \_Tp >, 2422
  - std::negate< \_Tp >, 2527
  - std::negate< void >, 2528
  - std::pointer\_to\_unary\_function< \_Arg, \_Result >, 2754
  - std::unary\_function< \_Arg, \_Result >, 3138
  - std::unary\_negate< \_Predicate >, 3139
- Arithmetic Function Object Classes, 311
- array, 3354, 3355
- Array creation functions, 281
  - make\_array, 282
  - to\_array, 282
- assertions.h, 3575
- assign
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 816–819
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1483, 1484
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1451–1453
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1566–1572
  - std::deque< \_Tp, \_Alloc >, 2039, 2040
  - std::error\_condition, 2116
  - std::forward\_list< \_Tp, \_Alloc >, 2146, 2147
  - std::list< \_Tp, \_Alloc >, 2333, 2334
  - std::vector< \_Tp, \_Alloc >, 3294, 3295
  - std::vector< bool, \_Alloc >, 3315, 3317
- assoc\_container.hpp, 3610
- assoc\_laguerre
  - Mathematical Special Functions, 193
  - TR1 Mathematical Special Functions, 248
- assoc\_laguerref
  - Mathematical Special Functions, 193
- assoc\_laguerrel
  - Mathematical Special Functions, 194
- assoc\_legendre
  - Mathematical Special Functions, 194
  - TR1 Mathematical Special Functions, 248
- assoc\_legendref
  - Mathematical Special Functions, 195
- assoc\_legendrel
  - Mathematical Special Functions, 195
- Associative, 121
- assume\_aligned
  - Memory, 319
- async
  - Futures, 114
- at
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 819, 820
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1485–1487
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1572, 1573
  - std::deque< \_Tp, \_Alloc >, 2040, 2041
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2386
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3174
  - std::vector< \_Tp, \_Alloc >, 3295, 3296
  - std::vector< bool, \_Alloc >, 3317
- ate
  - std::basic\_fstream< \_CharT, \_Traits >, 1155
  - std::basic\_ifstream< \_CharT, \_Traits >, 1198
  - std::basic\_ios< \_CharT, \_Traits >, 1221
  - std::basic\_iostream< \_CharT, \_Traits >, 1267
  - std::basic\_istream< \_CharT, \_Traits >, 1304
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2893
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1344
  - std::basic\_ofstream< \_CharT, \_Traits >, 1378
  - std::basic\_ostream< \_CharT, \_Traits >, 1408
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2921
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1442
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1712



- std::ios\_base, [2266](#)
- atomic, [3356](#)
- atomic\_base.h, [3377](#)
- atomic\_bool
  - Atomics, [104](#)
- ATOMIC\_BOOL\_LOCK\_FREE
  - Atomics, [103](#)
- atomic\_char
  - Atomics, [104](#)
- atomic\_char16\_t
  - Atomics, [104](#)
- atomic\_char32\_t
  - Atomics, [104](#)
- atomic\_compare\_exchange\_strong
  - Pointer Abstractions, [329](#)
- atomic\_compare\_exchange\_strong\_explicit
  - Pointer Abstractions, [330](#)
- atomic\_compare\_exchange\_weak
  - Pointer Abstractions, [331](#)
- atomic\_compare\_exchange\_weak\_explicit
  - Pointer Abstractions, [332](#)
- atomic\_exchange
  - Pointer Abstractions, [333](#)
- atomic\_exchange\_explicit
  - Pointer Abstractions, [333](#), [334](#)
- atomic\_futex.h, [3378](#)
- atomic\_int
  - Atomics, [104](#)
- atomic\_int16\_t
  - Atomics, [104](#)
- atomic\_int32\_t
  - Atomics, [104](#)
- atomic\_int64\_t
  - Atomics, [104](#)
- atomic\_int8\_t
  - Atomics, [105](#)
- atomic\_int\_fast16\_t
  - Atomics, [105](#)
- atomic\_int\_fast32\_t
  - Atomics, [105](#)
- atomic\_int\_fast64\_t
  - Atomics, [105](#)
- atomic\_int\_fast8\_t
  - Atomics, [105](#)
- atomic\_int\_least16\_t
  - Atomics, [105](#)
- atomic\_int\_least32\_t
  - Atomics, [105](#)
- atomic\_int\_least64\_t
  - Atomics, [106](#)
- atomic\_int\_least8\_t
  - Atomics, [106](#)
- atomic\_intmax\_t
  - Atomics, [106](#)
- atomic\_intptr\_t
  - Atomics, [106](#)
- atomic\_is\_lock\_free
  - Pointer Abstractions, [334](#)
- atomic\_llong
  - Atomics, [106](#)
- atomic\_load
  - Pointer Abstractions, [335](#)
- atomic\_load\_explicit
  - Pointer Abstractions, [335](#), [336](#)
- atomic\_lockfree\_defines.h, [3378](#)
- atomic\_long
  - Atomics, [106](#)
- atomic\_ptrdiff\_t
  - Atomics, [106](#)
- atomic\_schar
  - Atomics, [107](#)
- atomic\_short
  - Atomics, [107](#)
- atomic\_size\_t
  - Atomics, [107](#)
- atomic\_store
  - Pointer Abstractions, [336](#)
- atomic\_store\_explicit
  - Pointer Abstractions, [337](#)
- atomic\_timed\_wait.h, [3379](#)
- atomic\_uchar
  - Atomics, [107](#)
- atomic\_uint
  - Atomics, [107](#)
- atomic\_uint16\_t
  - Atomics, [107](#)
- atomic\_uint32\_t
  - Atomics, [107](#)
- atomic\_uint64\_t
  - Atomics, [108](#)
- atomic\_uint8\_t
  - Atomics, [108](#)
- atomic\_uint\_fast16\_t
  - Atomics, [108](#)
- atomic\_uint\_fast32\_t
  - Atomics, [108](#)
- atomic\_uint\_fast64\_t
  - Atomics, [108](#)
- atomic\_uint\_fast8\_t
  - Atomics, [108](#)
- atomic\_uint\_least16\_t
  - Atomics, [108](#)
- atomic\_uint\_least32\_t
  - Atomics, [109](#)
- atomic\_uint\_least64\_t
  - Atomics, [109](#)
- atomic\_uint\_least8\_t
  - Atomics, [109](#)



- atomic\_uintmax\_t
  - Atomics, 109
- atomic\_uintptr\_t
  - Atomics, 109
- atomic\_ullong
  - Atomics, 109
- atomic\_ulong
  - Atomics, 109
- atomic\_ushort
  - Atomics, 110
- atomic\_wait.h, 3379
- atomic\_wchar\_t
  - Atomics, 110
- atomic\_word.h, 3788
- atomicity.h, 3605
- Atomics, 99
  - atomic\_bool, 104
  - ATOMIC\_BOOL\_LOCK\_FREE, 103
  - atomic\_char, 104
  - atomic\_char16\_t, 104
  - atomic\_char32\_t, 104
  - atomic\_int, 104
  - atomic\_int16\_t, 104
  - atomic\_int32\_t, 104
  - atomic\_int64\_t, 104
  - atomic\_int8\_t, 105
  - atomic\_int\_fast16\_t, 105
  - atomic\_int\_fast32\_t, 105
  - atomic\_int\_fast64\_t, 105
  - atomic\_int\_fast8\_t, 105
  - atomic\_int\_least16\_t, 105
  - atomic\_int\_least32\_t, 105
  - atomic\_int\_least64\_t, 106
  - atomic\_int\_least8\_t, 106
  - atomic\_intmax\_t, 106
  - atomic\_intptr\_t, 106
  - atomic\_llong, 106
  - atomic\_long, 106
  - atomic\_ptrdiff\_t, 106
  - atomic\_schar, 107
  - atomic\_short, 107
  - atomic\_size\_t, 107
  - atomic\_uchar, 107
  - atomic\_uint, 107
  - atomic\_uint16\_t, 107
  - atomic\_uint32\_t, 107
  - atomic\_uint64\_t, 108
  - atomic\_uint8\_t, 108
  - atomic\_uint\_fast16\_t, 108
  - atomic\_uint\_fast32\_t, 108
  - atomic\_uint\_fast64\_t, 108
  - atomic\_uint\_fast8\_t, 108
  - atomic\_uint\_least16\_t, 108
  - atomic\_uint\_least32\_t, 109
  - atomic\_uint\_least64\_t, 109
  - atomic\_uint\_least8\_t, 109
  - atomic\_uintmax\_t, 109
  - atomic\_uintptr\_t, 109
  - atomic\_ullong, 109
  - atomic\_ulong, 109
  - atomic\_ushort, 110
  - atomic\_wchar\_t, 110
  - kill\_dependency, 110
  - memory\_order, 110
- auto\_ptr
  - std::auto\_ptr< \_Tp >, 1073
- auto\_ptr.h, 3360
- awk
  - std::regex\_constants, 715
- b
  - std::extreme\_value\_distribution< \_RealType >, 2125
  - std::weibull\_distribution< \_RealType >, 3346
- back
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 820
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1488
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1574
  - std::deque< \_Tp, \_Alloc >, 2041
  - std::list< \_Tp, \_Alloc >, 2334
  - std::queue< \_Tp, \_Sequence >, 2776
  - std::vector< \_Tp, \_Alloc >, 3296
  - std::vector< bool, \_Alloc >, 3318
- back\_insert\_iterator
  - std::back\_insert\_iterator< \_Container >, 1078
- back\_inserter
  - Iterators, 171
- backward\_warning.h, 3360
- bad
  - std::basic\_fstream< \_CharT, \_Traits >, 1120
  - std::basic\_ifstream< \_CharT, \_Traits >, 1171
  - std::basic\_ios< \_CharT, \_Traits >, 1210
  - std::basic\_iostream< \_CharT, \_Traits >, 1233
  - std::basic\_istream< \_CharT, \_Traits >, 1279
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2870
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1318
  - std::basic\_ofstream< \_CharT, \_Traits >, 1356
  - std::basic\_ostream< \_CharT, \_Traits >, 1389
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2903
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1421
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1678
- badbit
  - std::basic\_fstream< \_CharT, \_Traits >, 1155
  - std::basic\_ifstream< \_CharT, \_Traits >, 1198

- std::basic\_ios< \_CharT, \_Traits >, 1221
- std::basic\_iostream< \_CharT, \_Traits >, 1267
- std::basic\_istream< \_CharT, \_Traits >, 1305
- std::basic\_istream< \_CharT, \_Traits >::sentry, 2893
- std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1344
- std::basic\_ofstream< \_CharT, \_Traits >, 1378
- std::basic\_ostream< \_CharT, \_Traits >, 1408
- std::basic\_ostream< \_CharT, \_Traits >::sentry, 2921
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1442
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1712
- std::ios\_base, 2266
- balanced\_quicksort.h, 3724
- barrier, 3362
- base
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 967
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_UContainer >, 981
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2058
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2243
  - std::reverse\_iterator< \_Iterator >, 2835
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 2970
- Base and Implementation Classes, 272
  - \_Opcode, 272
- Base and Policy Classes, 140–142
- base.h, 3725
- basefield
  - std::basic\_fstream< \_CharT, \_Traits >, 1156
  - std::basic\_ifstream< \_CharT, \_Traits >, 1198
  - std::basic\_ios< \_CharT, \_Traits >, 1221
  - std::basic\_iostream< \_CharT, \_Traits >, 1268
  - std::basic\_istream< \_CharT, \_Traits >, 1305
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2893
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1344
  - std::basic\_ofstream< \_CharT, \_Traits >, 1378
  - std::basic\_ostream< \_CharT, \_Traits >, 1408
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2921
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1442
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1713
  - std::ios\_base, 2266
- basic
  - std::regex\_constants, 715
- basic\_file.h, 3788
- basic\_filebuf
  - std::basic\_filebuf< \_CharT, \_Traits >, 1093
- basic\_fstream
  - std::basic\_fstream< \_CharT, \_Traits >, 1118, 1119
- basic\_ifstream
  - std::basic\_ifstream< \_CharT, \_Traits >, 1169, 1170
- basic\_ios
  - std::basic\_ios< \_CharT, \_Traits >, 1209
- basic\_ios.h, 3379
- basic\_ios.tcc, 3379
- basic\_iostream
  - std::basic\_iostream< \_CharT, \_Traits >, 1232
- basic\_istream
  - std::basic\_istream< \_CharT, \_Traits >, 1278
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2870
- basic\_istreamstream
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1317
- basic\_iterator.h, 3726
- basic\_ofstream
  - std::basic\_ofstream< \_CharT, \_Traits >, 1355, 1356
- basic\_ostream
  - std::basic\_ostream< \_CharT, \_Traits >, 1388
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2903
- basic\_ostringstream
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1420
- basic\_regex
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1448–1450
- basic\_streambuf
  - std::basic\_streambuf< \_CharT, \_Traits >, 1460
- basic\_string
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1554–1560
- basic\_string.h, 3380
- basic\_string.tcc, 3382
- basic\_stringbuf
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1653
- basic\_stringstream
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1677
- before
  - \_\_gnu\_debug::type\_info, 3132
  - std::type\_info, 3133
- before\_begin
  - std::forward\_list< \_Tp, \_Alloc >, 2147
- beg
  - std::basic\_fstream< \_CharT, \_Traits >, 1156
  - std::basic\_ifstream< \_CharT, \_Traits >, 1198
  - std::basic\_ios< \_CharT, \_Traits >, 1221
  - std::basic\_iostream< \_CharT, \_Traits >, 1268
  - std::basic\_istream< \_CharT, \_Traits >, 1305
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2893
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1344

- std::basic\_ofstream< \_CharT, \_Traits >, 1378
- std::basic\_ostream< \_CharT, \_Traits >, 1408
- std::basic\_ostream< \_CharT, \_Traits >::sentry, 2921
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1442
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1713
- std::ios\_base, 2266
- begin
  - \_\_gnu\_cxx::Temporary\_buffer< \_ForwardIterator, \_Tp >, 1019
  - \_\_gnu\_cxx::versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 821
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 3048
  - \_\_gnu\_parallel::PseudoSequence< \_Tp, \_DifferenceTp >, 948
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 2854
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 3117
  - Filesystem, 162
  - Numeric Arrays, 222
  - std, 588
  - std::Temporary\_buffer< \_ForwardIterator, \_Tp >, 1021
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1574
  - std::deque< \_Tp, \_Alloc >, 2041
  - std::forward\_list< \_Tp, \_Alloc >, 2148
  - std::initializer\_list< \_E >, 2249
  - std::list< \_Tp, \_Alloc >, 2334
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2386
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2411
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2481
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2509
  - std::set< \_Key, \_Compare, \_Alloc >, 2936
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3175
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3205
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3233
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3259, 3261
  - std::vector< \_Tp, \_Alloc >, 3297
  - std::vector< bool, \_Alloc >, 3318
- Bernoulli Distributions, 234
  - operator<<, 235
  - operator>>, 235, 237
- bernoulli\_distribution
  - std::bernoulli\_distribution, 1718
- beta
  - Mathematical Special Functions, 195
- std::gamma\_distribution< \_RealType >, 2181
- TR1 Mathematical Special Functions, 249
- betaf
  - Mathematical Special Functions, 196
- betal
  - Mathematical Special Functions, 196
- bin\_search\_tree.hpp, 3610
- binary
  - std::basic\_fstream< \_CharT, \_Traits >, 1156
  - std::basic\_ifstream< \_CharT, \_Traits >, 1198
  - std::basic\_ios< \_CharT, \_Traits >, 1221
  - std::basic\_iostream< \_CharT, \_Traits >, 1268
  - std::basic\_istream< \_CharT, \_Traits >, 1305
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2893
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1344
  - std::basic\_ofstream< \_CharT, \_Traits >, 1378
  - std::basic\_ostream< \_CharT, \_Traits >, 1409
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2921
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1442
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1713
  - std::ios\_base, 2266
- Binary Search, 81
  - binary\_search, 82
  - equal\_range, 83
  - lower\_bound, 84
  - upper\_bound, 85
- binary\_heap.hpp, 3614
- binary\_heap\_const\_iterator\_
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1739
- binary\_heap\_point\_const\_iterator\_
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1743
- binary\_search
  - Binary Search, 82
- bind
  - Binder Classes, 313
- bind1st
  - Binder Classes, 313
- bind2nd
  - Binder Classes, 313
- Binder Classes, 311
  - bind, 313
  - bind1st, 313
  - bind2nd, 313
- binders.h, 3383
- binomial\_heap.hpp, 3627
- binomial\_heap\_base.hpp, 3628
- bit, 3362
- Bit manipulation, 177
- bitmap\_allocator.h, 3605

- `_BALLOC_ALIGN_BYTES`, 3606
- `bitset`, 3538
  - `std::bitset< _Nb >`, 1763, 1764
- `bool_set`, 3765
  - `std::tr2::bool_set`, 1771
- `bool_set.tcc`, 3766
- `boolalpha`
  - `std`, 589
  - `std::basic_fstream< _CharT, _Traits >`, 1156
  - `std::basic_ifstream< _CharT, _Traits >`, 1199
  - `std::basic_ios< _CharT, _Traits >`, 1222
  - `std::basic_iostream< _CharT, _Traits >`, 1268
  - `std::basic_istream< _CharT, _Traits >`, 1305
  - `std::basic_istream< _CharT, _Traits >::sentry`, 2893
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1345
  - `std::basic_ofstream< _CharT, _Traits >`, 1378
  - `std::basic_ostream< _CharT, _Traits >`, 1409
  - `std::basic_ostream< _CharT, _Traits >::sentry`, 2921
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1442
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1713
  - `std::ios_base`, 2267
- Boolean Operations Classes, 314
- `boost_concept_check.h`, 3383
- Branch-Based, 139
- `branch_policy.hpp`, 3628
- `bucket`
  - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 981
- `bucket_count`
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3176
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3206
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3234
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3261
- c
  - `std::queue< _Tp, _Sequence >`, 2777
- `c++0x_warning.h`, 3384
- `c++allocator.h`, 3788
- `c++config.h`, 3789
- `c++io.h`, 3796
- `c++locale.h`, 3796
- `c++locale_internal.h`, 3796
- `c_str`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 821
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1575
- `cache_line_size`
  - `__gnu_parallel::Settings`, 1010
- `call_once`
  - `Mutexes`, 117
  - `std::once_flag`, 2684
- `capacity`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 821
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1488
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1575
  - `std::vector< _Tp, _Alloc >`, 3297
  - `std::vector< bool, _Alloc >`, 3318
- `cassert`, 3539
- `cast.h`, 3606
- `category`
  - `std::error_code`, 2114
  - `std::error_condition`, 2116
  - `std::locale`, 2350
- `cbefore_begin`
  - `std::forward_list< _Tp, _Alloc >`, 2148
- `cbegin`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 821
  - `std`, 589
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1575
  - `std::deque< _Tp, _Alloc >`, 2042
  - `std::forward_list< _Tp, _Alloc >`, 2148
  - `std::list< _Tp, _Alloc >`, 2334
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, 2386
  - `std::match_results< _Bi_iter, _Alloc >`, 2411
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 2481
  - `std::multiset< _Key, _Compare, _Alloc >`, 2509
  - `std::set< _Key, _Compare, _Alloc >`, 2936
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3176
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3206
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3234
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3261
  - `std::vector< _Tp, _Alloc >`, 3297
  - `std::vector< bool, _Alloc >`, 3318
- `cc_hash_max_collision_check_resize_trigger`
  - `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >`, 1777
- `cc_hash_max_collision_check_resize_trigger_imp.hpp`, 3646
- `cc_hash_table`
  - `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`, 1781–1784
- `cc_ht_map.hpp`, 3629

- ccomplex, [3539](#)
- cctype, [3540](#)
- ceil
  - chrono.h, [3389](#)
  - Time, [366](#), [367](#)
- cend
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [821](#)
  - std, [589](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1576](#)
  - std::deque< \_Tp, \_Alloc >, [2042](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2148](#)
  - std::list< \_Tp, \_Alloc >, [2335](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2386](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2412](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [2481](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [2509](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [2936](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3176](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3206](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3234](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3262](#)
  - std::vector< \_Tp, \_Alloc >, [3297](#)
  - std::vector< bool, \_Alloc >, [3318](#)
- cerr
  - std, [646](#)
- cerrno, [3540](#)
- cfenv, [3540](#), [3541](#)
- cfloat, [3541](#)
- char\_traits.h, [3384](#)
- char\_type
  - std::\_\_ctype\_abstract\_base< \_CharT >, [751](#)
  - std::\_\_cxx11::collate< \_CharT >, [1857](#)
  - std::\_\_cxx11::collate\_byname< \_CharT >, [1862](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1207](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1459](#)
  - std::ctype< char >, [1926](#)
  - std::ctype< wchar\_t >, [1960](#)
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [2294](#)
  - std::messages< \_CharT >, [2430](#)
  - std::money\_get< \_CharT, \_InIter >, [2441](#)
  - std::money\_put< \_CharT, \_OutIter >, [2445](#)
  - std::moneypunct< \_CharT, \_Intl >, [2450](#)
  - std::num\_get< \_CharT, \_InIter >, [2546](#)
  - std::num\_put< \_CharT, \_OutIter >, [2561](#)
  - std::numpunct< \_CharT >, [2673](#)
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [2687](#)
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [2690](#)
  - std::time\_get< \_CharT, \_InIter >, [3063](#)
  - std::time\_put< \_CharT, \_OutIter >, [3085](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3330](#)
- charconv, [3541](#)
- charconv.h, [3385](#)
- chars\_format
  - std, [580](#)
- checkers.h, [3726](#)
- chrono, [3543](#), [3546](#)
- chrono.h, [3385](#)
- ceil, [3389](#)
- chrono\_io.h, [3389](#)
- cin
  - std, [646](#)
- cinttypes, [3546](#)
- ciso646, [3546](#)
- clamp
  - Sorting, [62](#), [63](#)
- classic
  - std::locale, [2353](#)
- classic\_table
  - std::ctype< char >, [1927](#)
  - std::ctype\_byname< char >, [1993](#)
- clear
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [822](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1120](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1171](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1210](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1233](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1279](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2870](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1318](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1356](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1389](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2904](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1421](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1576](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1678](#)
  - std::deque< \_Tp, \_Alloc >, [2042](#)
  - std::error\_condition, [2116](#)
  - std::experimental::fundamentals\_v1::any, [1065](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2148](#)
  - std::list< \_Tp, \_Alloc >, [2335](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2387](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [2481](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [2509](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [2936](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2075](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3177](#)

- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3207
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3235
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3262
- std::vector< \_Tp, \_Alloc >, 3297
- std::vector< bool, \_Alloc >, 3318
- climits, 3547
- clocale, 3547
- clock\_cast
  - Time, 367
- clog
  - std, 646
- close
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2087
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2991
  - std::basic\_filebuf< \_CharT, \_Traits >, 1094
  - std::basic\_fstream< \_CharT, \_Traits >, 1120
  - std::basic\_ifstream< \_CharT, \_Traits >, 1171
  - std::basic\_ofstream< \_CharT, \_Traits >, 1357
- cmath, 3547, 3558, 3559
- cmp\_fn
  - \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >, 3094
- cmp\_fn\_imps.hpp, 3629
- code
  - std::regex\_error, 2815
- codecvt, 3561
- codecvt.h, 3393
- codecvt\_specializations.h, 3607
- collate
  - std::\_\_cxx11::collate< \_CharT >, 1857, 1858
  - std::locale, 2356
  - std::regex\_constants, 715
- combine
  - std::locale, 2353
- common\_reference\_t
  - Metaprogramming, 351
- common\_type\_t
  - Metaprogramming, 351
- comp\_ellint\_1
  - Mathematical Special Functions, 196
  - TR1 Mathematical Special Functions, 249
- comp\_ellint\_1f
  - Mathematical Special Functions, 197
- comp\_ellint\_1l
  - Mathematical Special Functions, 197
- comp\_ellint\_2
  - Mathematical Special Functions, 197
  - TR1 Mathematical Special Functions, 249
- comp\_ellint\_2f
  - Mathematical Special Functions, 198
- comp\_ellint\_2l
  - Mathematical Special Functions, 198
- comp\_ellint\_3
  - Mathematical Special Functions, 198
  - TR1 Mathematical Special Functions, 249
- comp\_ellint\_3f
  - Mathematical Special Functions, 199
- comp\_ellint\_3l
  - Mathematical Special Functions, 199
- compare, 3800
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 822–824
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1488–1494
  - std::\_\_cxx11::collate< \_CharT >, 1858
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1576–1581
  - std::sub\_match< \_Bilter >, 3035
- compare\_three\_way\_result\_t
  - std, 579
- Comparison Classes, 314
- compatibility.h, 3797
- compiletime\_settings.h, 3726
  - \_GLIBCXX\_CALL, 3726
  - \_GLIBCXX\_PARALLEL\_ASSERTIONS, 3727
  - \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_L1, 3727
  - \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_TLB, 3727
  - \_GLIBCXX\_SCALE\_DOWN\_FPU, 3727
  - \_GLIBCXX\_VERBOSE\_LEVEL, 3727
- complex, 3562, 3566
  - std::complex< \_Tp >, 1867
  - std::complex< double >, 1869
  - std::complex< float >, 1871
  - std::complex< long double >, 1874
- Complex Numbers, 177
  - abs, 181
  - arg, 181
  - conj, 181
  - cos, 181
  - cosh, 181
  - exp, 181
  - fabs, 181
  - log, 181
  - log10, 182
  - norm, 182
  - operator<=, 185
  - operator>=, 185
  - operator+, 183
  - operator+=, 183
  - operator-, 183, 184
  - operator=, 184
  - operator/, 184
  - operator/=, 184

operator=, [185](#)  
 operator==, [185](#)  
 operator\*, [182](#)  
 operator\*==, [182](#)  
 polar, [185](#)  
 pow, [186](#)  
 sin, [186](#)  
 sinh, [186](#)  
 sqrt, [187](#)  
 tan, [187](#)  
 tanh, [187](#)  
 complex.h, [3567](#)  
 compose1  
     SGL, [153](#)  
 compose2  
     SGL, [153](#)  
 concept\_check.h, [3393](#)  
 concepts, [3567](#)  
 concurrence.h, [3607](#)  
 Concurrency, [111](#)  
 cond\_dealtor.hpp, [3633](#)  
 cond\_key\_dtor\_entry\_dealtor.hpp, [3630](#)  
 Condition Variables, [111](#)  
     cv\_status, [112](#)  
 condition\_variable, [3567](#)  
 conditional\_t  
     Metaprogramming, [351](#)  
 conf\_hyperg  
     Mathematical Special Functions, [199](#)  
     std::tr1, [724](#)  
 conf\_hypergfl  
     Mathematical Special Functions, [199](#)  
     std::tr1, [725](#)  
 conf\_hypergl  
     Mathematical Special Functions, [200](#)  
     std::tr1, [725](#)  
 conj  
     Complex Numbers, [181](#)  
 Const-propagating wrapper, [282](#)  
 const\_iterator  
     \_\_gnu\_pbds::trie\_string\_access\_traits< String,  
         Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >,  
         [3117](#)  
     std::set< \_Key, \_Compare, \_Alloc >, [2931](#)  
     std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
         loc >, [3170](#)  
     std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
         \_Alloc >, [3200](#)  
     std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
         loc >, [3229](#)  
     std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
         >, [3255](#)  
 const\_iterator.hpp, [3614](#), [3615](#)  
 const\_local\_iterator  
     std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
         loc >, [3170](#)  
     std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
         \_Alloc >, [3201](#)  
     std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
         loc >, [3229](#)  
     std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
         >, [3255](#)  
 const\_pointer  
     \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator<  
         Value\_Type, Entry, Simple, \_Alloc >, [1738](#)  
     \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator<  
         Value\_Type, Entry, Simple, \_Alloc >, [1742](#)  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator<  
         Node, \_Alloc >, [2305](#)  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator<  
         Node, \_Alloc >, [2309](#)  
     std::allocator\_traits< \_Alloc >, [1032](#)  
     std::allocator\_traits< allocator< \_Tp > >, [1038](#)  
     std::allocator\_traits< allocator< void > >, [1047](#)  
     std::allocator\_traits< pmr::polymorphic\_allocator<  
         \_Tp > >, [1055](#)  
     std::set< \_Key, \_Compare, \_Alloc >, [2931](#)  
     std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
         loc >, [3170](#)  
     std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
         \_Alloc >, [3201](#)  
     std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
         loc >, [3229](#)  
     std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
         >, [3255](#)  
 const\_pointer\_cast  
     Pointer Abstractions, [337](#)  
     std, [589](#)  
 const\_reference  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator<  
         Node, Const\_Iterator, Iterator, \_Alloc >, [1723](#)  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator<  
         Node, Const\_Iterator, Iterator, \_Alloc >, [1728](#)  
     \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator<  
         Value\_Type, Entry, Simple, \_Alloc >, [1738](#)  
     \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator<  
         Value\_Type, Entry, Simple, \_Alloc >, [1742](#)  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator<  
         Node, \_Alloc >, [2305](#)  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator<  
         Node, \_Alloc >, [2309](#)  
     std::set< \_Key, \_Compare, \_Alloc >, [2931](#)  
     std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
         loc >, [3170](#)  
     std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
         \_Alloc >, [3201](#)  
     std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
         loc >, [3229](#)



- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3256](#)
- const\_reverse\_iterator
  - std::set< \_Key, \_Compare, \_Alloc >, [2931](#)
- const\_void\_pointer
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [734](#)
  - std::allocator\_traits< \_Alloc >, [1032](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1038](#)
  - std::allocator\_traits< allocator< void > >, [1047](#)
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, [1056](#)
- constant0
  - SGL, [153](#)
- constant1
  - SGL, [153](#)
- constant2
  - SGL, [154](#)
- construct
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [736](#)
  - std::allocator\_traits< \_Alloc >, [1034](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1042](#)
  - std::allocator\_traits< allocator< void > >, [1051](#)
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, [1059](#), [1060](#)
- constructor\_destructor\_fn\_imps.hpp, [3630](#)
- constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp, [3630](#)
- constructor\_destructor\_store\_hash\_fn\_imps.hpp, [3630](#)
- constructors\_destructor\_fn\_imps.hpp, [3615](#), [3616](#)
- container\_base\_dispatch.hpp, [3633](#)
- container\_type
  - std::back\_insert\_iterator< \_Container >, [1077](#)
  - std::front\_insert\_iterator< \_Container >, [2163](#)
  - std::insert\_iterator< \_Container >, [2253](#)
- Containers, [120](#), [139](#)
- contains
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2387](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [2482](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [2509](#), [2510](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [2936](#), [2937](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3177](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3207](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3235](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3262](#)
- converted
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, [3349](#)
- copy
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [825](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1494](#)
  - Mutating, [21](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1582](#)
- copy\_backward
  - Mutating, [21](#)
- copy\_if
  - Mutating, [22](#)
- copy\_n
  - Mutating, [22](#)
  - SGL, [154](#)
- copy\_options
  - Filesystem, [161](#)
  - Filesystem TS, [279](#)
- copyable\_function
  - std::copyable\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, [1906](#), [1907](#)
- copyfmt
  - std::basic\_fstream< \_CharT, \_Traits >, [1120](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1171](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1210](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1233](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1279](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2870](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1318](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1357](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1389](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2904](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1423](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1678](#)
- cos
  - Complex Numbers, [181](#)
  - math.h, [3700](#)
- cosh
  - Complex Numbers, [181](#)
  - math.h, [3701](#)
- count
  - Non-Mutating, [45](#)
  - std::bitset< \_Nb >, [1765](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2387](#), [2388](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [2482](#), [2483](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [2510](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [2937](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2075](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3178](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3208](#)



- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3236
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3263
- count\_if
  - Non-Mutating, 46
- count\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1011
- cout
  - std, 646
- cow\_string.h, 3393
- cpp\_type\_traits.h, 3394
- cpu\_defines.h, 3797
- cpyfunc\_impl.h, 3394
- crbegin
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 825
  - std, 589
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1583
  - std::deque< \_Tp, \_Alloc >, 2042
  - std::list< \_Tp, \_Alloc >, 2335
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2388
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2483
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2510
  - std::set< \_Key, \_Compare, \_Alloc >, 2938
  - std::vector< \_Tp, \_Alloc >, 3298
  - std::vector< bool, \_Alloc >, 3319
- cref
  - std::reference\_wrapper< \_Tp >, 2814
- cregex\_token\_iterator
  - Regular Expressions, 255
- crend
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 825
  - std, 590
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1583
  - std::deque< \_Tp, \_Alloc >, 2042
  - std::list< \_Tp, \_Alloc >, 2335
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2388
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2483
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2511
  - std::set< \_Key, \_Compare, \_Alloc >, 2938
  - std::vector< \_Tp, \_Alloc >, 3298
  - std::vector< bool, \_Alloc >, 3319
- csetjmp, 3568
- cshift
  - Numeric Arrays, 222
- csignal, 3568
- cstdalign, 3569
- cstdarg, 3569
- cstdbool, 3569, 3570
- cstddef, 3570
- cstdint, 3570
- cstdio, 3571
- cstdlib, 3571
- cstring, 3572
- csub\_match
  - Regular Expressions, 255
- ctgmth, 3572
- ctime, 3573
- ctype
  - std::ctype< char >, 1926
  - std::ctype< wchar\_t >, 1960, 1961
  - std::locale, 2356
- ctype\_base.h, 3797
- ctype\_inline.h, 3798
- cuchar, 3573
- cur
  - std::basic\_fstream< \_CharT, \_Traits >, 1156
  - std::basic\_ifstream< \_CharT, \_Traits >, 1199
  - std::basic\_ios< \_CharT, \_Traits >, 1222
  - std::basic\_iostream< \_CharT, \_Traits >, 1268
  - std::basic\_istream< \_CharT, \_Traits >, 1305
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2894
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1345
  - std::basic\_ofstream< \_CharT, \_Traits >, 1379
  - std::basic\_ostream< \_CharT, \_Traits >, 1409
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2921
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1443
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1713
  - std::ios\_base, 2267
- curr\_symbol
  - std::moneypunct< \_CharT, \_Intl >, 2452
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2459
- current\_exception
  - Exceptions, 129
- cv\_status
  - Condition Variables, 112
- cwchar, 3573, 3574
- cwctype, 3574
- cxxabi.h, 3801
  - \_\_cxa\_demangle, 3802
- cxxabi\_forced.h, 3395
- cxxabi\_init\_exception.h, 3395
- cxxabi\_tweaks.h, 3798
- cyl\_bessel\_i
  - Mathematical Special Functions, 200
  - TR1 Mathematical Special Functions, 249
- cyl\_bessel\_if
  - Mathematical Special Functions, 200
- cyl\_bessel\_il
  - Mathematical Special Functions, 201
- cyl\_bessel\_j

- Mathematical Special Functions, [201](#)
- TR1 Mathematical Special Functions, [249](#)
- cyl\_bessel\_jf
  - Mathematical Special Functions, [201](#)
- cyl\_bessel\_jl
  - Mathematical Special Functions, [202](#)
- cyl\_bessel\_k
  - Mathematical Special Functions, [202](#)
  - TR1 Mathematical Special Functions, [249](#)
- cyl\_bessel\_kf
  - Mathematical Special Functions, [202](#)
- cyl\_bessel\_kl
  - Mathematical Special Functions, [203](#)
- cyl\_neumann
  - Mathematical Special Functions, [203](#)
  - TR1 Mathematical Special Functions, [250](#)
- cyl\_neumannf
  - Mathematical Special Functions, [203](#)
- cyl\_neumannl
  - Mathematical Special Functions, [204](#)
- data
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [826](#)
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, [1495](#)
  - `std`, [590](#), [591](#)
  - `std::basic_string< _CharT, _Traits, _Alloc >`, [1583](#), [1584](#)
  - `std::vector< _Tp, _Alloc >`, [3298](#)
  - `std::vector< bool, _Alloc >`, [3319](#)
- Data parallel extensions, [294](#)
- Data Structure Type, [145](#)
- date\_order
  - `std::time_get< _CharT, _InIter >`, [3063](#)
  - `std::time_get_byname< _CharT, _InIter >`, [3074](#)
- days
  - Time, [365](#)
- deallocate
  - `__gnu_cxx::__alloc_traits< _Alloc, typename >`, [737](#)
  - `std::allocator_traits< _Alloc >`, [1035](#)
  - `std::allocator_traits< allocator< _Tp > >`, [1043](#)
  - `std::allocator_traits< allocator< void > >`, [1051](#), [1052](#)
  - `std::allocator_traits< pmr::polymorphic_allocator< _Tp > >`, [1060](#)
- debug.h, [3575](#)
- debug\_allocator.h, [3608](#)
- debug\_fn\_imps.hpp, [3616](#)–[3618](#)
- debug\_map\_base.hpp, [3634](#)
- debug\_no\_store\_hash\_fn\_imps.hpp, [3631](#)
- debug\_store\_hash\_fn\_imps.hpp, [3631](#)
- dec
  - `std`, [591](#)
- `std::basic_fstream< _CharT, _Traits >`, [1156](#)
- `std::basic_ifstream< _CharT, _Traits >`, [1199](#)
- `std::basic_ios< _CharT, _Traits >`, [1222](#)
- `std::basic_iostream< _CharT, _Traits >`, [1268](#)
- `std::basic_istream< _CharT, _Traits >`, [1305](#)
- `std::basic_istream< _CharT, _Traits >::sentry`, [2894](#)
- `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, [1345](#)
- `std::basic_ofstream< _CharT, _Traits >`, [1379](#)
- `std::basic_ostream< _CharT, _Traits >`, [1409](#)
- `std::basic_ostream< _CharT, _Traits >::sentry`, [2921](#)
- `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, [1443](#)
- `std::basic_stringstream< _CharT, _Traits, _Alloc >`, [1713](#)
- `std::ios_base`, [2267](#)
- decay\_t
  - Metaprogramming, [351](#)
- decimal, [3591](#)
- Decimal Floating-Point Arithmetic, [187](#)
- decimal128
  - `std::decimal::decimal128`, [2015](#)
- decimal32
  - `std::decimal::decimal32`, [2017](#)
- decimal32\_to\_long\_long
  - `std::decimal`, [690](#)
- decimal64
  - `std::decimal::decimal64`, [2018](#)
- decimal\_point
  - `std::moneypunct< _CharT, _Intl >`, [2452](#)
  - `std::moneypunct_byname< _CharT, _Intl >`, [2459](#)
  - `std::numpunct< _CharT >`, [2674](#)
  - `std::numpunct_byname< _CharT >`, [2678](#)
- declare\_no\_pointers
  - Pointer Safety and Garbage Collection, [344](#)
- declare\_reachable
  - Pointer Safety and Garbage Collection, [344](#)
- declval
  - Utilities, [299](#)
- default\_delete
  - `std::default_delete< _Tp >`, [2020](#)
  - `std::default_delete< _Tp[] >`, [2021](#)
- default\_error\_condition
  - `std::error_category`, [2112](#)
  - `std::error_code`, [2114](#)
- default\_sentinel
  - `std`, [646](#)
- defaultfloat
  - `std`, [591](#)
- defer\_lock
  - Mutexes, [118](#)
- denorm\_absent
  - `std`, [580](#)
- denorm\_indeterminate

- std, 580
- denorm\_min
  - std::numeric\_limits< \_Tp >, 2573
  - std::numeric\_limits< bool >, 2579
  - std::numeric\_limits< char >, 2584
  - std::numeric\_limits< char16\_t >, 2589
  - std::numeric\_limits< char32\_t >, 2595
  - std::numeric\_limits< double >, 2600
  - std::numeric\_limits< float >, 2605
  - std::numeric\_limits< int >, 2610
  - std::numeric\_limits< long >, 2615
  - std::numeric\_limits< long double >, 2621
  - std::numeric\_limits< long long >, 2626
  - std::numeric\_limits< short >, 2631
  - std::numeric\_limits< signed char >, 2637
  - std::numeric\_limits< unsigned char >, 2642
  - std::numeric\_limits< unsigned int >, 2647
  - std::numeric\_limits< unsigned long >, 2652
  - std::numeric\_limits< unsigned long long >, 2658
  - std::numeric\_limits< unsigned short >, 2663
  - std::numeric\_limits< wchar\_t >, 2668
- denorm\_present
  - std, 580
- densities
  - std::piecewise\_constant\_distribution< \_RealType >, 2743
  - std::piecewise\_linear\_distribution< \_RealType >, 2747
- Deprecated List, 1
- deque, 3600, 3601
  - std::deque< \_Tp, \_Alloc >, 2033–2035
- deque.tcc, 3395
- destroy
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 737
  - std::allocator\_traits< \_Alloc >, 1035
  - std::allocator\_traits< allocator< \_Tp > >, 1043, 1044
  - std::allocator\_traits< allocator< void > >, 1052
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, 1061
- destroying\_delete
  - std, 646
- detected\_or
  - Detection idiom, 285
- detected\_or\_t
  - Detection idiom, 285
- detected\_t
  - Detection idiom, 285
- Detection idiom, 284
  - \_\_cpp\_lib\_experimental\_detect, 285
  - detected\_or, 285
  - detected\_or\_t, 285
  - detected\_t, 285
  - is\_detected, 285
  - is\_detected\_convertible, 285
  - is\_detected\_convertible\_v, 285
  - is\_detected\_exact, 285
  - is\_detected\_exact\_v, 286
  - is\_detected\_v, 286
  - void\_t, 285
- Diagnostics, 123
  - generic\_category, 124
  - make\_error\_code, 124
  - make\_error\_condition, 124
  - operator<<, 125
  - operator<=>, 125
  - operator==, 125, 126
  - system\_category, 126
- difference\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, 1723
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, 1728
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 1738
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 1742
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_< Node, \_Alloc >, 2305
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, \_Alloc >, 2309
  - std::allocator\_traits< \_Alloc >, 1033
  - std::allocator\_traits< allocator< \_Tp > >, 1038, 1039
  - std::allocator\_traits< allocator< void > >, 1047
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, 1056
  - std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2292
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 2294
  - std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 2300
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 2687
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 2690
  - std::pointer\_traits< \_Ptr >, 2754
  - std::pointer\_traits< \_Tp \* >, 2756
  - std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 2795
  - std::set< \_Key, \_Compare, \_Alloc >, 2932
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3170
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3201
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3229
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3256
- digits

- std::\_\_numeric\_limits\_base, 788
- std::numeric\_limits< \_Tp >, 2574
- std::numeric\_limits< bool >, 2580
- std::numeric\_limits< char >, 2585
- std::numeric\_limits< char16\_t >, 2590
- std::numeric\_limits< char32\_t >, 2596
- std::numeric\_limits< double >, 2601
- std::numeric\_limits< float >, 2606
- std::numeric\_limits< int >, 2611
- std::numeric\_limits< long >, 2616
- std::numeric\_limits< long double >, 2622
- std::numeric\_limits< long long >, 2627
- std::numeric\_limits< short >, 2632
- std::numeric\_limits< signed char >, 2638
- std::numeric\_limits< unsigned char >, 2643
- std::numeric\_limits< unsigned int >, 2648
- std::numeric\_limits< unsigned long >, 2653
- std::numeric\_limits< unsigned long long >, 2659
- std::numeric\_limits< unsigned short >, 2664
- std::numeric\_limits< wchar\_t >, 2669
- digits10
  - std::\_\_numeric\_limits\_base, 788
  - std::numeric\_limits< \_Tp >, 2575
  - std::numeric\_limits< bool >, 2580
  - std::numeric\_limits< char >, 2585
  - std::numeric\_limits< char16\_t >, 2590
  - std::numeric\_limits< char32\_t >, 2596
  - std::numeric\_limits< double >, 2601
  - std::numeric\_limits< float >, 2606
  - std::numeric\_limits< int >, 2611
  - std::numeric\_limits< long >, 2616
  - std::numeric\_limits< long double >, 2622
  - std::numeric\_limits< long long >, 2627
  - std::numeric\_limits< short >, 2632
  - std::numeric\_limits< signed char >, 2638
  - std::numeric\_limits< unsigned char >, 2643
  - std::numeric\_limits< unsigned int >, 2648
  - std::numeric\_limits< unsigned long >, 2653
  - std::numeric\_limits< unsigned long long >, 2659
  - std::numeric\_limits< unsigned short >, 2664
  - std::numeric\_limits< wchar\_t >, 2669
- direct\_mask\_range\_hashing\_imp.hpp, 3636
- direct\_mod\_range\_hashing\_imp.hpp, 3636
- directory\_options
  - Filesystem, 161
  - Filesystem TS, 279
- discard
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2058
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2243
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2321
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 2426
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 2970
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3040
- discard\_block\_engine
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2057, 2058
- distance
  - SGL, 154
  - std, 591
- do\_allocate
  - std::pmr::monotonic\_buffer\_resource, 2466
  - std::pmr::unsynchronized\_pool\_resource, 3278
- do\_always\_noconv
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 1801
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1807
  - std::codecvt< char, char, mbstate\_t >, 1816
  - std::codecvt< char16\_t, char, mbstate\_t >, 1825
  - std::codecvt< char32\_t, char, mbstate\_t >, 1834
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1843
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1852
- do\_compare
  - std::\_\_cxx11::collate< \_CharT >, 1858
- do\_curr\_symbol
  - std::moneypunct< \_CharT, \_Intl >, 2452
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2460
- do\_date\_order
  - std::time\_get< \_CharT, \_InIter >, 3063
  - std::time\_get\_byname< \_CharT, \_InIter >, 3074
- do\_deallocate
  - std::pmr::monotonic\_buffer\_resource, 2466
  - std::pmr::unsynchronized\_pool\_resource, 3278
- do\_decimal\_point
  - std::moneypunct< \_CharT, \_Intl >, 2452
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2460
  - std::numpunct< \_CharT >, 2674
  - std::numpunct\_byname< \_CharT >, 2679
- do\_encoding
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 1801
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1807
  - std::codecvt< char, char, mbstate\_t >, 1816
  - std::codecvt< char16\_t, char, mbstate\_t >, 1825
  - std::codecvt< char32\_t, char, mbstate\_t >, 1834
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1843
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1852
- do\_falsename
  - std::numpunct< \_CharT >, 2675
  - std::numpunct\_byname< \_CharT >, 2679

- do\_frac\_digits
  - std::moneypunct< \_CharT, \_Intl >, 2453
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2460
- do\_get
  - std::messages< \_CharT >, 2431
  - std::messages\_byname< \_CharT >, 2433
  - std::money\_get< \_CharT, \_Inlter >, 2442
  - std::num\_get< \_CharT, \_Inlter >, 2546–2552
  - std::time\_get< \_CharT, \_Inlter >, 3064
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3074
- do\_get\_date
  - std::time\_get< \_CharT, \_Inlter >, 3064
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3075
- do\_get\_monthname
  - std::time\_get< \_CharT, \_Inlter >, 3065
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3075
- do\_get\_time
  - std::time\_get< \_CharT, \_Inlter >, 3066
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3076
- do\_get\_weekday
  - std::time\_get< \_CharT, \_Inlter >, 3066
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3077
- do\_get\_year
  - std::time\_get< \_CharT, \_Inlter >, 3067
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3077
- do\_grouping
  - std::moneypunct< \_CharT, \_Intl >, 2453
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2461
  - std::numpunct< \_CharT >, 2675
  - std::numpunct\_byname< \_CharT >, 2679
- do\_hash
  - std::\_\_cxx11::collate< \_CharT >, 1859
- do\_in
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 1802
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1807, 1808
  - std::codecvt< char, char, mbstate\_t >, 1816
  - std::codecvt< char16\_t, char, mbstate\_t >, 1825
  - std::codecvt< char32\_t, char, mbstate\_t >, 1834
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1843
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1852
- do\_is
  - std::\_\_ctype\_abstract\_base< \_CharT >, 751
  - std::ctype< \_CharT >, 1912
  - std::ctype< char >, 1927, 1928
  - std::ctype< wchar\_t >, 1961, 1962
  - std::ctype\_byname< \_CharT >, 1980
  - std::ctype\_byname< char >, 1993, 1994
- do\_is\_equal
  - std::pmr::monotonic\_buffer\_resource, 2466
  - std::pmr::unsynchronized\_pool\_resource, 3278
- do\_length
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 1802
- std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1808
- std::codecvt< char, char, mbstate\_t >, 1817
- std::codecvt< char16\_t, char, mbstate\_t >, 1825, 1826
- std::codecvt< char32\_t, char, mbstate\_t >, 1834, 1835
- std::codecvt< wchar\_t, char, mbstate\_t >, 1844
- std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1853
- do\_max\_length
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 1802
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1808
  - std::codecvt< char, char, mbstate\_t >, 1817
  - std::codecvt< char16\_t, char, mbstate\_t >, 1826
  - std::codecvt< char32\_t, char, mbstate\_t >, 1835
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1844
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1853
- do\_narrow
  - std::\_\_ctype\_abstract\_base< \_CharT >, 752
  - std::ctype< \_CharT >, 1913
  - std::ctype< char >, 1929, 1930, 1932
  - std::ctype< wchar\_t >, 1963, 1964
  - std::ctype\_byname< \_CharT >, 1980, 1981
  - std::ctype\_byname< char >, 1995, 1996
- do\_neg\_format
  - std::moneypunct< \_CharT, \_Intl >, 2453
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2461
- do\_negative\_sign
  - std::moneypunct< \_CharT, \_Intl >, 2453
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2461
- do\_out
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, 743
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 1802
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1808, 1809
  - std::codecvt< char, char, mbstate\_t >, 1817
  - std::codecvt< char16\_t, char, mbstate\_t >, 1826
  - std::codecvt< char32\_t, char, mbstate\_t >, 1835
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1844
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1853
- do\_pos\_format
  - std::moneypunct< \_CharT, \_Intl >, 2454
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2461
- do\_positive\_sign
  - std::moneypunct< \_CharT, \_Intl >, 2454
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2462
- do\_put
  - std::money\_put< \_CharT, \_Outlter >, 2446
  - std::num\_put< \_CharT, \_Outlter >, 2562–2566

- std::time\_put< \_CharT, \_Outiter >, 3085
- std::time\_put\_byname< \_CharT, \_Outiter >, 3089
- do\_scan\_is
  - std::\_\_ctype\_abstract\_base< \_CharT >, 753
  - std::ctype< \_CharT >, 1914
  - std::ctype< char >, 1933
  - std::ctype< wchar\_t >, 1964, 1965
  - std::ctype\_byname< \_CharT >, 1981
  - std::ctype\_byname< char >, 1997
- do\_scan\_not
  - std::\_\_ctype\_abstract\_base< \_CharT >, 753
  - std::ctype< \_CharT >, 1914
  - std::ctype< char >, 1934
  - std::ctype< wchar\_t >, 1965, 1966
  - std::ctype\_byname< \_CharT >, 1982
  - std::ctype\_byname< char >, 1998
- do\_thousands\_sep
  - std::moneypunct< \_CharT, \_Intl >, 2454
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2462
  - std::numpunct< \_CharT >, 2675
  - std::numpunct\_byname< \_CharT >, 2679
- do\_tolower
  - std::\_\_ctype\_abstract\_base< \_CharT >, 754
  - std::ctype< \_CharT >, 1915
  - std::ctype< char >, 1935–1937
  - std::ctype< wchar\_t >, 1966–1968
  - std::ctype\_byname< \_CharT >, 1982, 1983
  - std::ctype\_byname< char >, 1999, 2000
- do\_toupper
  - std::\_\_ctype\_abstract\_base< \_CharT >, 755
  - std::ctype< \_CharT >, 1916
  - std::ctype< char >, 1937–1939
  - std::ctype< wchar\_t >, 1968, 1969
  - std::ctype\_byname< \_CharT >, 1983, 1984
  - std::ctype\_byname< char >, 2000, 2001
- do\_transform
  - std::\_\_cxx11::collate< \_CharT >, 1859
- do\_truename
  - std::numpunct< \_CharT >, 2675
  - std::numpunct\_byname< \_CharT >, 2680
- do\_unshift
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 1802
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1809
  - std::codecvt< char, char, mbstate\_t >, 1818
  - std::codecvt< char16\_t, char, mbstate\_t >, 1827
  - std::codecvt< char32\_t, char, mbstate\_t >, 1836
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1845
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1853
- do\_widen
  - std::\_\_ctype\_abstract\_base< \_CharT >, 756
  - std::ctype< \_CharT >, 1917
  - std::ctype< char >, 1940, 1941
- std::ctype< wchar\_t >, 1970, 1971
- std::ctype\_byname< \_CharT >, 1984
- std::ctype\_byname< char >, 2002, 2003
- duration\_cast
  - Time, 367
- Dynamic Bitset., 134
  - operator!=, 135
  - operator<<, 136
  - operator<=, 136
  - operator>, 136
  - operator>>, 137
  - operator>=, 136
  - operator-, 135
  - operator&, 135
  - operator^, 137
  - operator|, 137
- dynamic\_bitset, 3766
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2073, 2074
- dynamic\_bitset.tcc, 3767
- dynamic\_pointer\_cast
  - Pointer Abstractions, 338
  - std, 592
- e\_pos
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 2854
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 3118
- e\_type
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 2854
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 3117
- eback
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2087
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2992
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3011
  - std::basic\_filebuf< \_CharT, \_Traits >, 1094
  - std::basic\_streambuf< \_CharT, \_Traits >, 1460
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1654
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3331
- ECMAScript
  - std::regex\_constants, 715
- egptr
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2087
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2992
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3011
  - std::basic\_filebuf< \_CharT, \_Traits >, 1094
  - std::basic\_streambuf< \_CharT, \_Traits >, 1460
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1654
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3331



- egrep
  - std::regex\_constants, [715](#)
- element\_type
  - std::auto\_ptr< \_Tp >, [1072](#)
  - std::pointer\_traits< \_Ptr >, [2754](#)
  - std::pointer\_traits< \_Tp \* >, [2756](#)
  - std::shared\_ptr< \_Tp >, [2960](#)
- ellint\_1
  - Mathematical Special Functions, [204](#)
  - TR1 Mathematical Special Functions, [250](#)
- ellint\_1f
  - Mathematical Special Functions, [204](#)
- ellint\_1l
  - Mathematical Special Functions, [205](#)
- ellint\_2
  - Mathematical Special Functions, [205](#)
  - TR1 Mathematical Special Functions, [250](#)
- ellint\_2f
  - Mathematical Special Functions, [206](#)
- ellint\_2l
  - Mathematical Special Functions, [206](#)
- ellint\_3
  - Mathematical Special Functions, [206](#)
  - TR1 Mathematical Special Functions, [250](#)
- ellint\_3f
  - Mathematical Special Functions, [207](#)
- ellint\_3l
  - Mathematical Special Functions, [207](#)
- emplace
  - std::deque< \_Tp, \_Alloc >, [2042](#)
  - std::list< \_Tp, \_Alloc >, [2335](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2388](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [2483](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [2511](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [2938](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3178](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3208](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3236](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3263](#)
  - std::vector< \_Tp, \_Alloc >, [3298](#)
  - std::vector< bool, \_Alloc >, [3319](#)
- emplace\_after
  - std::forward\_list< \_Tp, \_Alloc >, [2148](#)
- emplace\_front
  - std::forward\_list< \_Tp, \_Alloc >, [2149](#)
- emplace\_hint
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2389](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [2484](#)
- std::multiset< \_Key, \_Compare, \_Alloc >, [2511](#)
- std::set< \_Key, \_Compare, \_Alloc >, [2938](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3180](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3209](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3237](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3264](#)
- empty
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [826](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1495](#)
  - \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [1788](#)
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [2194](#)
- std, [592](#)
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1584](#)
- std::deque< \_Tp, \_Alloc >, [2043](#)
- std::experimental::fundamentals\_v1::any, [1065](#)
- std::forward\_list< \_Tp, \_Alloc >, [2149](#)
- std::list< \_Tp, \_Alloc >, [2335](#)
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2389](#)
- std::match\_results< \_Bi\_iter, \_Alloc >, [2412](#)
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [2484](#)
- std::multiset< \_Key, \_Compare, \_Alloc >, [2512](#)
- std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [2767](#)
- std::queue< \_Tp, \_Sequence >, [2776](#)
- std::set< \_Key, \_Compare, \_Alloc >, [2939](#)
- std::stack< \_Tp, \_Sequence >, [2986](#)
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2075](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3180](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3209](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3237](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3264](#)
- std::vector< \_Tp, \_Alloc >, [3298](#)
- std::vector< bool, \_Alloc >, [3319](#)
- enable\_if\_t
  - Metaprogramming, [351](#)
- enable\_special\_members.h, [3397](#)
- enc\_filebuf.h, [3608](#)
- end

[\\_\\_gnu\\_cxx::Temporary\\_buffer< \\_ForwardIterator, \\_Tp >](#), 1019  
[\\_\\_gnu\\_cxx::versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >](#), 826  
[\\_\\_gnu\\_cxx::temporary\\_buffer< \\_ForwardIterator, \\_Tp >](#), 3048  
[\\_\\_gnu\\_parallel::PseudoSequence< \\_Tp, \\_DifferenceTp >](#), 948  
[\\_\\_gnu\\_pbds::detail::trie\\_policy\\_base< Node\\_Cltr, Node\\_ltr, \\_ATraits, \\_Alloc >](#), 3113  
[\\_\\_gnu\\_pbds::sample\\_trie\\_access\\_traits](#), 2855  
[\\_\\_gnu\\_pbds::trie\\_string\\_access\\_traits< String, Min\\_E\\_Val, Max\\_E\\_Val, Reverse, \\_Alloc >](#), 3118  
 Filesystem, 162  
 Numeric Arrays, 223  
 std, 592, 593  
[std::Temporary\\_buffer< \\_ForwardIterator, \\_Tp >](#), 1021  
[std::basic\\_fstream< \\_CharT, \\_Traits >](#), 1156  
[std::basic\\_ifstream< \\_CharT, \\_Traits >](#), 1199  
[std::basic\\_ios< \\_CharT, \\_Traits >](#), 1222  
[std::basic\\_iostream< \\_CharT, \\_Traits >](#), 1268  
[std::basic\\_istream< \\_CharT, \\_Traits >](#), 1306  
[std::basic\\_istream< \\_CharT, \\_Traits >::sentry](#), 2894  
[std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), 1345  
[std::basic\\_ofstream< \\_CharT, \\_Traits >](#), 1379  
[std::basic\\_ostream< \\_CharT, \\_Traits >](#), 1409  
[std::basic\\_ostream< \\_CharT, \\_Traits >::sentry](#), 2921  
[std::basic\\_ostringstream< \\_CharT, \\_Traits, \\_Alloc >](#), 1443  
[std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >](#), 1584, 1585  
[std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), 1713  
[std::deque< \\_Tp, \\_Alloc >](#), 2043  
[std::forward\\_list< \\_Tp, \\_Alloc >](#), 2149  
[std::initializer\\_list< \\_E >](#), 2249  
[std::ios\\_base](#), 2267  
[std::list< \\_Tp, \\_Alloc >](#), 2336  
[std::map< \\_Key, \\_Tp, \\_Compare, \\_Alloc >](#), 2390  
[std::match\\_results< \\_Bi\\_iter, \\_Alloc >](#), 2412  
[std::multimap< \\_Key, \\_Tp, \\_Compare, \\_Alloc >](#), 2484  
[std::multiset< \\_Key, \\_Compare, \\_Alloc >](#), 2512  
[std::set< \\_Key, \\_Compare, \\_Alloc >](#), 2939  
[std::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#), 3180, 3181  
[std::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#), 3209, 3210  
[std::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#), 3237, 3238  
[std::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#), 3264, 3265  
[std::vector< \\_Tp, \\_Alloc >](#), 3298, 3299  
[std::vector< bool, \\_Alloc >](#), 3319  
 endl  
 std, 593  
 ends  
 std, 593  
[entry\\_cmp.hpp](#), 3618  
[entry\\_list\\_fn\\_imps.hpp](#), 3631  
[entry\\_metadata\\_base.hpp](#), 3640  
[entry\\_pred.hpp](#), 3618  
 eof  
[std::basic\\_fstream< \\_CharT, \\_Traits >](#), 1122  
[std::basic\\_ifstream< \\_CharT, \\_Traits >](#), 1173  
[std::basic\\_ios< \\_CharT, \\_Traits >](#), 1211  
[std::basic\\_iostream< \\_CharT, \\_Traits >](#), 1234  
[std::basic\\_istream< \\_CharT, \\_Traits >](#), 1279  
[std::basic\\_istream< \\_CharT, \\_Traits >::sentry](#), 2871  
[std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), 1319  
[std::basic\\_ofstream< \\_CharT, \\_Traits >](#), 1357  
[std::basic\\_ostream< \\_CharT, \\_Traits >](#), 1389  
[std::basic\\_ostream< \\_CharT, \\_Traits >::sentry](#), 2904  
[std::basic\\_ostringstream< \\_CharT, \\_Traits, \\_Alloc >](#), 1423  
[std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), 1679  
 eofbit  
[std::basic\\_fstream< \\_CharT, \\_Traits >](#), 1157  
[std::basic\\_ifstream< \\_CharT, \\_Traits >](#), 1199  
[std::basic\\_ios< \\_CharT, \\_Traits >](#), 1222  
[std::basic\\_iostream< \\_CharT, \\_Traits >](#), 1269  
[std::basic\\_istream< \\_CharT, \\_Traits >](#), 1306  
[std::basic\\_istream< \\_CharT, \\_Traits >::sentry](#), 2894  
[std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), 1345  
[std::basic\\_ofstream< \\_CharT, \\_Traits >](#), 1379  
[std::basic\\_ostream< \\_CharT, \\_Traits >](#), 1409  
[std::basic\\_ostream< \\_CharT, \\_Traits >::sentry](#), 2922  
[std::basic\\_ostringstream< \\_CharT, \\_Traits, \\_Alloc >](#), 1443  
[std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), 1714  
[std::ios\\_base](#), 2267  
 ep\_ptr  
[\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >](#), 2087  
[\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >](#), 2992  
[\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#), 3012  
[std::basic\\_filebuf< \\_CharT, \\_Traits >](#), 1094  
[std::basic\\_streambuf< \\_CharT, \\_Traits >](#), 1461  
[std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#), 1654  
[std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >](#), 3331  
 epsilon



- std::numeric\_limits< \_Tp >, 2573
- std::numeric\_limits< bool >, 2579
- std::numeric\_limits< char >, 2584
- std::numeric\_limits< char16\_t >, 2589
- std::numeric\_limits< char32\_t >, 2595
- std::numeric\_limits< double >, 2600
- std::numeric\_limits< float >, 2605
- std::numeric\_limits< int >, 2610
- std::numeric\_limits< long >, 2615
- std::numeric\_limits< long double >, 2621
- std::numeric\_limits< long long >, 2626
- std::numeric\_limits< short >, 2631
- std::numeric\_limits< signed char >, 2637
- std::numeric\_limits< unsigned char >, 2642
- std::numeric\_limits< unsigned int >, 2647
- std::numeric\_limits< unsigned long >, 2652
- std::numeric\_limits< unsigned long long >, 2658
- std::numeric\_limits< unsigned short >, 2663
- std::numeric\_limits< wchar\_t >, 2668
- eq\_by\_less.hpp, 3634
- equal
  - Non-Mutating, 46, 47
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 2296
- equal\_range
  - Binary Search, 83
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2390, 2391
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2485, 2486
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2512, 2513
  - std::set< \_Key, \_Compare, \_Alloc >, 2939, 2940
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3181, 3182
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3210, 3211
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3238, 3239
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3265, 3267
- equally\_split.h, 3727
- equals
  - std::tr2::bool\_set, 1771
- equivalent
  - std::error\_category, 2112
- erase
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 826, 827
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1495, 1496
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1585, 1587, 1588
  - std::deque< \_Tp, \_Alloc >, 2043
  - std::list< \_Tp, \_Alloc >, 2336
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2392
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2486, 2487
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2514
  - std::set< \_Key, \_Compare, \_Alloc >, 2941, 2942
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3183, 3184
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3212, 3213
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3239–3241
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3269, 3270
  - std::vector< \_Tp, \_Alloc >, 3299
  - std::vector< bool, \_Alloc >, 3320
- erase\_after
  - std::chrono::tzdb\_list, 3136
  - std::forward\_list< \_Tp, \_Alloc >, 2150
- erase\_can\_throw
  - \_\_gnu\_pbds::container\_traits< Cntr >, 1899
- erase\_fn\_imps.hpp, 3618–3620
- erase\_if.h, 3397
- erase\_no\_store\_hash\_fn\_imps.hpp, 3631
- erase\_store\_hash\_fn\_imps.hpp, 3631
- error\_backref
  - std::regex\_constants, 711
- error\_badbrace
  - std::regex\_constants, 711
- error\_badrepeat
  - std::regex\_constants, 711
- error\_brace
  - std::regex\_constants, 711
- error\_brack
  - std::regex\_constants, 711
- error\_code
  - std::error\_code, 2114
- error\_collate
  - std::regex\_constants, 711
- error\_complexity
  - std::regex\_constants, 711
- error\_condition
  - std::error\_condition, 2115
- error\_constants.h, 3798
- error\_ctype
  - std::regex\_constants, 711
- error\_escape
  - std::regex\_constants, 712
- error\_paren
  - std::regex\_constants, 712
- error\_range
  - std::regex\_constants, 712
- error\_space
  - std::regex\_constants, 712
- error\_stack
  - std::regex\_constants, 712

- error\_type
  - std::regex\_constants, 710
- event
  - std::basic\_fstream< \_CharT, \_Traits >, 1118
  - std::basic\_ifstream< \_CharT, \_Traits >, 1169
  - std::basic\_ios< \_CharT, \_Traits >, 1209
  - std::basic\_iostream< \_CharT, \_Traits >, 1232
  - std::basic\_istream< \_CharT, \_Traits >, 1278
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2869
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1317
  - std::basic\_ofstream< \_CharT, \_Traits >, 1355
  - std::basic\_ostream< \_CharT, \_Traits >, 1388
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2902
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1420
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1677
  - std::ios\_base, 2260
- event\_callback
  - std::basic\_fstream< \_CharT, \_Traits >, 1117
  - std::basic\_ifstream< \_CharT, \_Traits >, 1168
  - std::basic\_ios< \_CharT, \_Traits >, 1207
  - std::basic\_iostream< \_CharT, \_Traits >, 1231
  - std::basic\_istream< \_CharT, \_Traits >, 1277
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2868
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1314
  - std::basic\_ofstream< \_CharT, \_Traits >, 1354
  - std::basic\_ostream< \_CharT, \_Traits >, 1387
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2901
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1419
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1675
  - std::ios\_base, 2259
- exception, 3803
- exception.h, 3397
- exception.hpp, 3653
- exception\_defines.h, 3398
- exception\_ptr.h, 3398
  - rethrow\_exception, 3398
- Exceptions, 127, 143
  - \_\_verbose\_terminate\_handler, 129
  - current\_exception, 129
  - fmtflags, 132
  - get\_terminate, 129
  - get\_unexpected, 129
  - make\_exception\_ptr, 130
  - rethrow\_exception, 130
  - rethrow\_if\_nested, 130
  - set\_terminate, 130
  - set\_unexpected, 131
  - terminate, 131
  - terminate\_handler, 128
  - throw\_with\_nested, 131
  - uncaught\_exception, 131
  - unexpected, 132
  - unexpected\_handler, 128
- exceptions
  - std::basic\_fstream< \_CharT, \_Traits >, 1122
  - std::basic\_ifstream< \_CharT, \_Traits >, 1173
  - std::basic\_ios< \_CharT, \_Traits >, 1211
  - std::basic\_iostream< \_CharT, \_Traits >, 1234
  - std::basic\_istream< \_CharT, \_Traits >, 1280
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2871
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1319
  - std::basic\_ofstream< \_CharT, \_Traits >, 1357, 1358
  - std::basic\_ostream< \_CharT, \_Traits >, 1390
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2904
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1423
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1679
- exclusive\_scan
  - Generalized Numeric operations, 8, 9
- exp
  - Complex Numbers, 181
  - math.h, 3701
- expected, 3602
- expint
  - Mathematical Special Functions, 207
  - TR1 Mathematical Special Functions, 250
- expintf
  - Mathematical Special Functions, 208
- expintl
  - Mathematical Special Functions, 208
- exponential\_distribution
  - std::exponential\_distribution< \_RealType >, 2121
- extc++.h, 3799
- extended
  - std::regex\_constants, 716
- Extensions, 133
- external\_load\_access
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1777
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2225
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, 2235
- extptr\_allocator.h, 3608
- fabs
  - Complex Numbers, 181
  - math.h, 3701
  - std, 593

- facet
  - std::locale::facet, [2128](#)
- fail
  - std::basic\_fstream< \_CharT, \_Traits >, [1123](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1174](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1212](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1235](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1280](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2872](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1320](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1358](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1390](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2905](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1424](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1680](#)
- failbit
  - std::basic\_fstream< \_CharT, \_Traits >, [1157](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1199](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1222](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1269](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1306](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2894](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1345](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1379](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1409](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2922](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1443](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1714](#)
  - std::ios\_base, [2267](#)
- failed
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [2691](#)
- false\_type
  - Metaprogramming, [351](#)
- falsename
  - std::numpunct< \_CharT >, [2676](#)
  - std::numpunct\_byname< \_CharT >, [2680](#)
- fd
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [2992](#)
- features.h, [3728](#)
  - \_GLIBCXX\_BAL\_QUICKSORT, [3728](#)
  - \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS, [3728](#)
  - \_GLIBCXX\_FIND\_EQUAL\_SPLIT, [3728](#)
  - \_GLIBCXX\_FIND\_GROWING\_BLOCKS, [3728](#)
  - \_GLIBCXX\_MERGESORT, [3729](#)
  - \_GLIBCXX\_QUICKSORT, [3729](#)
  - \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING, [3729](#)
  - \_GLIBCXX\_TREE\_FULL\_COPY, [3729](#)
  - \_GLIBCXX\_TREE\_INITIAL\_SPLITTING, [3729](#)
- fenv.h, [3679](#)
- file
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [2993](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3012](#)
- file\_time\_type
  - Filesystem, [161](#)
  - Filesystem TS, [279](#)
- file\_type
  - Filesystem, [162](#)
  - Filesystem TS, [279](#)
- filebuf
  - I/O, [165](#)
- Filesystem, [157](#)
  - begin, [162](#)
  - copy\_options, [161](#)
  - directory\_options, [161](#)
  - end, [162](#)
  - file\_time\_type, [161](#)
  - file\_type, [162](#)
  - perm\_options, [162](#)
  - perms, [162](#)
  - u8path, [162](#), [163](#)
- filesystem, [3679](#)
- Filesystem TS, [274](#)
  - copy\_options, [279](#)
  - directory\_options, [279](#)
  - file\_time\_type, [279](#)
  - file\_type, [279](#)
  - perms, [279](#)
- fill
  - Mutating, [23](#)
- std::basic\_fstream< \_CharT, \_Traits >, [1123](#)
- std::basic\_ifstream< \_CharT, \_Traits >, [1174](#)
- std::basic\_ios< \_CharT, \_Traits >, [1212](#)
- std::basic\_iostream< \_CharT, \_Traits >, [1235](#)
- std::basic\_istream< \_CharT, \_Traits >, [1281](#)
- std::basic\_istream< \_CharT, \_Traits >::sentry, [2872](#)
- std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1320](#)
- std::basic\_ofstream< \_CharT, \_Traits >, [1358](#), [1359](#)
- std::basic\_ostream< \_CharT, \_Traits >, [1391](#)
- std::basic\_ostream< \_CharT, \_Traits >::sentry, [2905](#)
- std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1424](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1680](#)
- fill\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1011](#)
- fill\_n
  - Mutating, [23](#)
- find

- `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 827–829
- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1497–1501
- Non-Mutating, 48
- `std::basic_string< _CharT, _Traits, _Alloc >`, 1588–1591
- `std::map< _Key, _Tp, _Compare, _Alloc >`, 2393, 2394
- `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 2488, 2489
- `std::multiset< _Key, _Compare, _Alloc >`, 2515, 2516
- `std::set< _Key, _Compare, _Alloc >`, 2942, 2943
- `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3184, 3185
- `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3213, 3214
- `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3241, 3243
- `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3270, 3271
- `find.h`, 3730
- `find_by_order`
  - `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`, 3097
  - `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, 3111
- `find_end`
  - Non-Mutating, 48, 49
- `find_first`
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2075
- `find_first_not_of`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 829, 830
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1501–1505
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1592–1594
- `find_first_of`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 831, 832
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1506–1510
  - Non-Mutating, 50
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1595–1597
- `find_fn_imps.hpp`, 3620, 3621
- `find_if`
  - Non-Mutating, 51
- `find_if_not`
  - Non-Mutating, 51
- `find_increasing_factor`
  - `__gnu_parallel::Settings`, 1011
- `find_initial_block_size`
  - `__gnu_parallel::Settings`, 1011
- `find_last_not_of`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 832–834
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1510–1514
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1598–1600
- `find_last_of`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 834, 835
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1514–1518
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1601–1603
- `find_maximum_block_size`
  - `__gnu_parallel::Settings`, 1011
- `find_next`
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2075
- `find_no_store_hash_fn_imps.hpp`, 3635
- `find_scale_factor`
  - `__gnu_parallel::Settings`, 1011
- `find_selectors.h`, 3730
- `find_sequential_search_size`
  - `__gnu_parallel::Settings`, 1011
- `find_store_hash_fn_imps.hpp`, 3631, 3632
- `first`
  - `__gnu_cxx::pair< _T1, _T2 >`, 2710
  - `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`, 899
  - `std::pair< _T1, _T2 >`, 2714
- `first_argument_type`
  - `__gnu_cxx::equal_to< _Tp >`, 2108
  - `__gnu_cxx::project1st< _Arg1, _Arg2 >`, 2769
  - `__gnu_cxx::project2nd< _Arg1, _Arg2 >`, 2770
  - `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >`, 876
  - `__gnu_parallel::_EqualTo< _T1, _T2 >`, 877
  - `__gnu_parallel::_Less< _T1, _T2 >`, 903
  - `__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >`, 904
  - `__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >`, 905
  - `__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >`, 931
  - `__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >`, 943
  - `std::binary_function< _Arg1, _Arg2, _Result >`, 1735
  - `std::binary_negate< _Predicate >`, 1745
  - `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 1882
  - `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`, 1883
  - `std::divides< _Tp >`, 2065
  - `std::divides< void >`, 2066
  - `std::equal_to< _Tp >`, 2109

- std::equal\_to< void >, 2111
- std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > >, 2703
- std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >, 2705
- std::greater< \_Tp >, 2197
- std::greater< void >, 2199
- std::greater\_equal< \_Tp >, 2200
- std::greater\_equal< void >, 2202
- std::less< \_Tp >, 2313
- std::less< void >, 2315
- std::less\_equal< \_Tp >, 2316
- std::less\_equal< void >, 2317
- std::logical\_and< \_Tp >, 2359
- std::logical\_and< void >, 2361
- std::logical\_or< \_Tp >, 2364
- std::logical\_or< void >, 2365
- std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 2419
- std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 2420
- std::minus< \_Tp >, 2434
- std::minus< void >, 2435
- std::modulus< \_Tp >, 2437
- std::modulus< void >, 2438
- std::multiplies< \_Tp >, 2499
- std::multiplies< void >, 2500
- std::not\_equal\_to< \_Tp >, 2539
- std::not\_equal\_to< void >, 2541
- std::owner\_less< shared\_ptr< \_Tp > >, 2704
- std::owner\_less< void >, 2705
- std::owner\_less< weak\_ptr< \_Tp > >, 2706
- std::plus< \_Tp >, 2750
- std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >, 2752
- first\_type
  - \_\_gnu\_cxx::pair< \_T1, \_T2 >, 2709
  - \_\_gnu\_parallel::iterator\_pair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 895, 896
  - std::pair< \_T1, \_T2 >, 2712
- fixed
  - std, 593
  - std::basic\_fstream< \_CharT, \_Traits >, 1157
  - std::basic\_ifstream< \_CharT, \_Traits >, 1200
  - std::basic\_ios< \_CharT, \_Traits >, 1223
  - std::basic\_iostream< \_CharT, \_Traits >, 1269
  - std::basic\_istream< \_CharT, \_Traits >, 1306
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2894
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1346
  - std::basic\_ofstream< \_CharT, \_Traits >, 1379
  - std::basic\_ostream< \_CharT, \_Traits >, 1410
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2922
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1443
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1714
- std::ios\_base, 2268
- flags
  - std::basic\_fstream< \_CharT, \_Traits >, 1123, 1124
  - std::basic\_ifstream< \_CharT, \_Traits >, 1174, 1175
  - std::basic\_ios< \_CharT, \_Traits >, 1212, 1213
  - std::basic\_iostream< \_CharT, \_Traits >, 1235, 1236
  - std::basic\_istream< \_CharT, \_Traits >, 1281
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2872
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1320, 1321
  - std::basic\_ofstream< \_CharT, \_Traits >, 1359
  - std::basic\_ostream< \_CharT, \_Traits >, 1391
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2906
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1425
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1453
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1680, 1681
  - std::ios\_base, 2261
- flat\_map, 3680
- flat\_set, 3680
- flip
  - std::bitset< \_Nb >, 1765
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2076
- float\_denorm\_style
  - std, 580
- float\_round\_style
  - std, 580
- floatfield
  - std::basic\_fstream< \_CharT, \_Traits >, 1157
  - std::basic\_ifstream< \_CharT, \_Traits >, 1200
  - std::basic\_ios< \_CharT, \_Traits >, 1223
  - std::basic\_iostream< \_CharT, \_Traits >, 1269
  - std::basic\_istream< \_CharT, \_Traits >, 1306
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2894
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1346
  - std::basic\_ofstream< \_CharT, \_Traits >, 1380
  - std::basic\_ostream< \_CharT, \_Traits >, 1410
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2922
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1444
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1714
  - std::ios\_base, 2268
- floor
  - Time, 368
- flush
  - std, 594
  - std::basic\_fstream< \_CharT, \_Traits >, 1124
  - std::basic\_iostream< \_CharT, \_Traits >, 1236
  - std::basic\_ofstream< \_CharT, \_Traits >, 1359

- std::basic\_ostream< \_CharT, \_Traits >, 1392
- std::basic\_ostream< \_CharT, \_Traits >::sentry, 2906
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1425
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1681
- fmtflags
  - Exceptions, 132
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2894
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2922
- for\_each
  - Non-Mutating, 52
- for\_each.h, 3730
- for\_each\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1011
- for\_each\_n
  - Non-Mutating, 52
- for\_each\_selectors.h, 3731
- format, 3680
  - std::filesystem::path, 2740
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2412, 2413
- format\_default
  - std::regex\_constants, 716
- format\_first\_only
  - std::regex\_constants, 716
- format\_no\_copy
  - std::regex\_constants, 716
- format\_sed
  - std::regex\_constants, 716
- formatfwd.h, 3398
- formatter.h, 3576
- forward
  - Utilities, 300
- forward\_as\_tuple
  - Utilities, 301
- forward\_list, 3680, 3681
  - std::forward\_list< \_Tp, \_Alloc >, 2143, 2144, 2146
- forward\_list.h, 3399
- forward\_list.tcc, 3399
- fpos
  - std::fpos< \_StateT >, 2159
- frac\_digits
  - std::moneypunct< \_CharT, \_Intl >, 2455
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2462
- frexp
  - std, 594
- from\_bytes
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, 3349, 3350
- from\_chars
  - std, 594
- front
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 836
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1519
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1604
  - std::chrono::tzdb\_list, 3136
  - std::deque< \_Tp, \_Alloc >, 2044
  - std::forward\_list< \_Tp, \_Alloc >, 2150, 2151
  - std::list< \_Tp, \_Alloc >, 2337
  - std::queue< \_Tp, \_Sequence >, 2776, 2777
  - std::vector< \_Tp, \_Alloc >, 3300
  - std::vector< bool, \_Alloc >, 3320
- front\_insert\_iterator
  - std::front\_insert\_iterator< \_Container >, 2164
- front\_inserter
  - Iterators, 171
- fs\_dir.h, 3400
- fs\_fwd.h, 3401, 3403
- fs\_ops.h, 3404, 3407
- fs\_path.h, 3409
- fstream, 3682
  - I/O, 165
- fstream.tcc, 3410
- funcref\_impl.h, 3410
- functexcept.h, 3410
- function
  - std::function< \_Res(\_ArgTypes...)>, 2166, 2167
- Function Objects, 307
  - mem\_fn, 308
- function\_ref
  - std::function\_ref< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 2172
- functional, 3683–3685
- functional\_hash.h, 3411
- functions.h, 3576
- funcwrap.h, 3412
- future, 3687
  - std::future< \_Res >, 2174
  - std::future< \_Res & >, 2176
  - std::future< void >, 2178
- future\_category
  - Futures, 115
- future\_errc
  - Futures, 114
- future\_status
  - Futures, 114
- Futures, 112
  - async, 114
  - future\_category, 115
  - future\_errc, 114
  - future\_status, 114
  - launch, 114
  - make\_error\_code, 115
  - make\_error\_condition, 115
  - swap, 115



- gamma\_distribution
  - std::gamma\_distribution< \_RealType >, 2181
- gbump
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2088
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2993
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3012
  - std::basic\_filebuf< \_CharT, \_Traits >, 1095
  - std::basic\_streambuf< \_CharT, \_Traits >, 1461
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1654
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3332
- gcd
  - std::experimental, 701
- gcount
  - std::basic\_fstream< \_CharT, \_Traits >, 1124
  - std::basic\_ifstream< \_CharT, \_Traits >, 1175
  - std::basic\_iostream< \_CharT, \_Traits >, 1236
  - std::basic\_istream< \_CharT, \_Traits >, 1282
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2873
  - std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 1321
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1681
- Generalized Numeric operations, 5
  - accumulate, 6, 7
  - adjacent\_difference, 7, 8
  - exclusive\_scan, 8, 9
  - inclusive\_scan, 9, 10
  - inner\_product, 11
  - iota, 12
  - partial\_sum, 12, 13
  - reduce, 13, 14
  - transform\_exclusive\_scan, 15
  - transform\_inclusive\_scan, 15, 16
  - transform\_reduce, 17, 18
- generate
  - Mutating, 24
- generate\_canonical
  - Random Number Generation, 233
- generate\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1011
- generate\_n
  - Mutating, 24
- generator, 3688
- generic\_category
  - Diagnostics, 124
- get
  - \_\_gnu\_parallel::Settings, 1010
  - std, 594–596
  - std::auto\_ptr< \_Tp >, 1074
  - std::basic\_fstream< \_CharT, \_Traits >, 1124–1126
  - std::basic\_ifstream< \_CharT, \_Traits >, 1175–1177
  - std::basic\_iostream< \_CharT, \_Traits >, 1236–1238
  - std::basic\_istream< \_CharT, \_Traits >, 1282–1284
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2873, 2874
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1321–1323
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1681–1683
  - std::future< \_Res >, 2174
  - std::future< \_Res & >, 2176
  - std::future< void >, 2178
  - std::money\_get< \_CharT, \_InIter >, 2442, 2443
  - std::num\_get< \_CharT, \_InIter >, 2553–2559
  - std::shared\_future< \_Res >, 2952
  - std::shared\_future< \_Res & >, 2954, 2955
  - std::shared\_future< void >, 2957
  - std::shared\_ptr< \_Tp >, 2966
  - std::time\_get< \_CharT, \_InIter >, 3067, 3068
  - std::time\_get\_byname< \_CharT, \_InIter >, 3078
  - std::unique\_ptr< \_Tp, \_Dp >, 3151
  - std::unique\_ptr< \_Tp[], \_Dp >, 3159
  - Utilities, 301
- get\_actual\_size
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, 2236
- get\_allocator
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 836
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1519
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1604
  - std::deque< \_Tp, \_Alloc >, 2044
  - std::forward\_list< \_Tp, \_Alloc >, 2151
  - std::list< \_Tp, \_Alloc >, 2337
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2394
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2413
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2489
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2516
  - std::set< \_Key, \_Compare, \_Alloc >, 2943
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2076
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3186
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3215
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3243
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3272
  - std::vector< \_Tp, \_Alloc >, 3300
  - std::vector< bool, \_Alloc >, 3320, 3321
- get\_child
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 936

- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >, [938](#)
- get\_comb\_hash\_fn
  - \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [1788](#)
- get\_comb\_probe\_fn
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [2194](#), [2195](#)
- get\_date
  - std::time\_get< \_CharT, \_Inlter >, [3069](#)
  - std::time\_get\_byname< \_CharT, \_Inlter >, [3079](#)
- get\_default\_resource
  - memory\_resource, [3706](#)
- get\_deleter
  - Pointer Abstractions, [338](#)
  - std::experimental, [701](#)
  - std::unique\_ptr< \_Tp, \_Dp >, [3152](#)
  - std::unique\_ptr< \_Tp[], \_Dp >, [3159](#)
- get\_eq\_fn
  - \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [1789](#)
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [2195](#)
- get\_hash\_fn
  - \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [1789](#)
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [2195](#)
- get\_id
  - std::this\_thread, [721](#)
- get\_l\_child
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1724](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1729](#)
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, [2698](#)
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, [2700](#)
- get\_load
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1777](#)
- get\_loads
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< Ex-ternal\_Load\_Access, Size\_Type >, [2225](#)
- \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, [2236](#)
- get\_metadata
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1724](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1729](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >, [936](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >, [939](#)
- get\_money
  - std, [596](#)
- get\_monthname
  - std::time\_get< \_CharT, \_Inlter >, [3069](#)
  - std::time\_get\_byname< \_CharT, \_Inlter >, [3080](#)
- get\_nearest\_larger\_size
  - \_\_gnu\_pbds::sample\_size\_policy, [2853](#)
- get\_nearest\_smaller\_size
  - \_\_gnu\_pbds::sample\_size\_policy, [2853](#)
- get\_new\_handler
  - std, [596](#)
- get\_new\_size
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, [2236](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [2848](#)
- get\_pointer\_safety
  - Pointer Safety and Garbage Collection, [344](#)
- get\_probe\_fn
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [2195](#)
- get\_r\_child
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1725](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1729](#)
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, [2698](#)
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, [2700](#)
- get\_resize\_policy
  - \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [1789](#)
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >



- [>](#), [2196](#)
- get\_size\_policy
  - [\\_\\_gnu\\_pbds::hash\\_standard\\_resize\\_policy< Size\\_Policy, Trigger\\_Policy, External\\_Size\\_Access, Size\\_Type >](#), [2237](#)
- get\_temporary\_buffer
  - [std](#), [596](#)
- get\_terminate
  - Exceptions, [129](#)
- get\_time
  - [std](#), [596](#)
  - [std::time\\_get< \\_CharT, \\_InIter >](#), [3070](#)
  - [std::time\\_get\\_byname< \\_CharT, \\_InIter >](#), [3080](#)
- get\_trigger\_policy
  - [\\_\\_gnu\\_pbds::hash\\_standard\\_resize\\_policy< Size\\_Policy, Trigger\\_Policy, External\\_Size\\_Access, Size\\_Type >](#), [2237](#)
- get\_unexpected
  - Exceptions, [129](#)
- get\_weekday
  - [std::time\\_get< \\_CharT, \\_InIter >](#), [3070](#)
  - [std::time\\_get\\_byname< \\_CharT, \\_InIter >](#), [3081](#)
- get\_year
  - [std::time\\_get< \\_CharT, \\_InIter >](#), [3071](#)
  - [std::time\\_get\\_byname< \\_CharT, \\_InIter >](#), [3081](#)
- getline
  - [std](#), [597](#), [598](#)
  - [std::basic\\_fstream< \\_CharT, \\_Traits >](#), [1127](#), [1128](#)
  - [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), [1178](#), [1179](#)
  - [std::basic\\_iostream< \\_CharT, \\_Traits >](#), [1239](#), [1240](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >](#), [1284](#), [1285](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >::sentry](#), [2875](#), [2877](#)
  - [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1324](#)
  - [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1684](#), [1685](#)
- getloc
  - [\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >](#), [2088](#)
  - [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >](#), [2993](#)
  - [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#), [3012](#)
  - [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [1095](#)
  - [std::basic\\_fstream< \\_CharT, \\_Traits >](#), [1128](#)
  - [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), [1179](#)
  - [std::basic\\_ios< \\_CharT, \\_Traits >](#), [1213](#)
  - [std::basic\\_iostream< \\_CharT, \\_Traits >](#), [1240](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >](#), [1285](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >::sentry](#), [2877](#)
  - [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1325](#)
  - [std::basic\\_ofstream< \\_CharT, \\_Traits >](#), [1360](#)
  - [std::basic\\_ostream< \\_CharT, \\_Traits >](#), [1392](#)
  - [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry](#), [2906](#)
- [std::basic\\_ostreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1425](#)
- [std::basic\\_regex< \\_Ch\\_type, \\_Rx\\_traits >](#), [1453](#)
- [std::basic\\_streambuf< \\_CharT, \\_Traits >](#), [1461](#)
- [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#), [1656](#)
- [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1685](#)
- [std::ios\\_base](#), [2261](#)
- [std::regex\\_traits< \\_Ch\\_type >](#), [2824](#)
- [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >](#), [3332](#)
- global
  - [std::locale](#), [2353](#)
- good
  - [std::basic\\_fstream< \\_CharT, \\_Traits >](#), [1128](#)
  - [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), [1179](#)
  - [std::basic\\_ios< \\_CharT, \\_Traits >](#), [1213](#)
  - [std::basic\\_iostream< \\_CharT, \\_Traits >](#), [1240](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >](#), [1286](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >::sentry](#), [2877](#)
  - [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1325](#)
  - [std::basic\\_ofstream< \\_CharT, \\_Traits >](#), [1360](#)
  - [std::basic\\_ostream< \\_CharT, \\_Traits >](#), [1392](#)
  - [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry](#), [2906](#)
  - [std::basic\\_ostreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1426](#)
  - [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1685](#)
- goodbit
  - [std::basic\\_fstream< \\_CharT, \\_Traits >](#), [1157](#)
  - [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), [1200](#)
  - [std::basic\\_ios< \\_CharT, \\_Traits >](#), [1223](#)
  - [std::basic\\_iostream< \\_CharT, \\_Traits >](#), [1269](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >](#), [1306](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >::sentry](#), [2895](#)
  - [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1346](#)
  - [std::basic\\_ofstream< \\_CharT, \\_Traits >](#), [1380](#)
  - [std::basic\\_ostream< \\_CharT, \\_Traits >](#), [1410](#)
  - [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry](#), [2923](#)
  - [std::basic\\_ostreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1444](#)
  - [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), [1714](#)
  - [std::ios\\_base](#), [2268](#)
- gp\_hash\_table
  - [\\_\\_gnu\\_pbds::gp\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Comb\\_Probe\\_Fn, Probe\\_Fn, Resize\\_Policy, Store\\_Hash, \\_Alloc >](#), [2187](#)–[2190](#)
- gp\_ht\_map.hpp, [3635](#)
- gptr
  - [\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >](#), [2088](#)
  - [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >](#), [2993](#)

- `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3013
- `std::basic_filebuf< _CharT, _Traits >`, 1095
- `std::basic_streambuf< _CharT, _Traits >`, 1462
- `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1656
- `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3332
- grep
  - `std::regex_constants`, 717
- grouping
  - `std::moneypunct< _CharT, _Intl >`, 2455
  - `std::moneypunct_byname< _CharT, _Intl >`, 2463
  - `std::numpunct< _CharT >`, 2676
  - `std::numpunct_byname< _CharT >`, 2680
- gslice
  - Numeric Arrays, 218
- `gslice.h`, 3412
- `gslice_array`
  - Numeric Arrays, 219
- `gslice_array.h`, 3412
- has\_denorm
  - `std::__numeric_limits_base`, 788
  - `std::numeric_limits< _Tp >`, 2575
  - `std::numeric_limits< bool >`, 2580
  - `std::numeric_limits< char >`, 2585
  - `std::numeric_limits< char16_t >`, 2590
  - `std::numeric_limits< char32_t >`, 2596
  - `std::numeric_limits< double >`, 2601
  - `std::numeric_limits< float >`, 2606
  - `std::numeric_limits< int >`, 2611
  - `std::numeric_limits< long >`, 2617
  - `std::numeric_limits< long double >`, 2622
  - `std::numeric_limits< long long >`, 2627
  - `std::numeric_limits< short >`, 2632
  - `std::numeric_limits< signed char >`, 2638
  - `std::numeric_limits< unsigned char >`, 2643
  - `std::numeric_limits< unsigned int >`, 2648
  - `std::numeric_limits< unsigned long >`, 2653
  - `std::numeric_limits< unsigned long long >`, 2659
  - `std::numeric_limits< unsigned short >`, 2664
  - `std::numeric_limits< wchar_t >`, 2669
- has\_denorm\_loss
  - `std::__numeric_limits_base`, 788
  - `std::numeric_limits< _Tp >`, 2575
  - `std::numeric_limits< bool >`, 2580
  - `std::numeric_limits< char >`, 2585
  - `std::numeric_limits< char16_t >`, 2591
  - `std::numeric_limits< char32_t >`, 2596
  - `std::numeric_limits< double >`, 2601
  - `std::numeric_limits< float >`, 2606
  - `std::numeric_limits< int >`, 2611
  - `std::numeric_limits< long >`, 2617
  - `std::numeric_limits< long double >`, 2622
  - `std::numeric_limits< long long >`, 2627
- `std::numeric_limits< short >`, 2633
- `std::numeric_limits< signed char >`, 2638
- `std::numeric_limits< unsigned char >`, 2643
- `std::numeric_limits< unsigned int >`, 2648
- `std::numeric_limits< unsigned long >`, 2654
- `std::numeric_limits< unsigned long long >`, 2659
- `std::numeric_limits< unsigned short >`, 2664
- `std::numeric_limits< wchar_t >`, 2669
- has\_facet
  - Locales, 174
  - `std::locale`, 2355
  - `std::locale::id`, 2239
- has\_infinity
  - `std::__numeric_limits_base`, 788
  - `std::numeric_limits< _Tp >`, 2575
  - `std::numeric_limits< bool >`, 2580
  - `std::numeric_limits< char >`, 2585
  - `std::numeric_limits< char16_t >`, 2591
  - `std::numeric_limits< char32_t >`, 2596
  - `std::numeric_limits< double >`, 2601
  - `std::numeric_limits< float >`, 2606
  - `std::numeric_limits< int >`, 2612
  - `std::numeric_limits< long >`, 2617
  - `std::numeric_limits< long double >`, 2622
  - `std::numeric_limits< long long >`, 2627
  - `std::numeric_limits< short >`, 2633
  - `std::numeric_limits< signed char >`, 2638
  - `std::numeric_limits< unsigned char >`, 2643
  - `std::numeric_limits< unsigned int >`, 2648
  - `std::numeric_limits< unsigned long >`, 2654
  - `std::numeric_limits< unsigned long long >`, 2659
  - `std::numeric_limits< unsigned short >`, 2664
  - `std::numeric_limits< wchar_t >`, 2669
- has\_quiet\_NaN
  - `std::__numeric_limits_base`, 789
  - `std::numeric_limits< _Tp >`, 2575
  - `std::numeric_limits< bool >`, 2580
  - `std::numeric_limits< char >`, 2585
  - `std::numeric_limits< char16_t >`, 2591
  - `std::numeric_limits< char32_t >`, 2596
  - `std::numeric_limits< double >`, 2601
  - `std::numeric_limits< float >`, 2606
  - `std::numeric_limits< int >`, 2612
  - `std::numeric_limits< long >`, 2617
  - `std::numeric_limits< long double >`, 2622
  - `std::numeric_limits< long long >`, 2627
  - `std::numeric_limits< short >`, 2633
  - `std::numeric_limits< signed char >`, 2638
  - `std::numeric_limits< unsigned char >`, 2643
  - `std::numeric_limits< unsigned int >`, 2648
  - `std::numeric_limits< unsigned long >`, 2654
  - `std::numeric_limits< unsigned long long >`, 2659
  - `std::numeric_limits< unsigned short >`, 2664
  - `std::numeric_limits< wchar_t >`, 2669

## has\_signaling\_NaN

std::\_\_numeric\_limits\_base, 789  
 std::numeric\_limits< \_Tp >, 2575  
 std::numeric\_limits< bool >, 2580  
 std::numeric\_limits< char >, 2585  
 std::numeric\_limits< char16\_t >, 2591  
 std::numeric\_limits< char32\_t >, 2596  
 std::numeric\_limits< double >, 2601  
 std::numeric\_limits< float >, 2606  
 std::numeric\_limits< int >, 2612  
 std::numeric\_limits< long >, 2617  
 std::numeric\_limits< long double >, 2622  
 std::numeric\_limits< long long >, 2627  
 std::numeric\_limits< short >, 2633  
 std::numeric\_limits< signed char >, 2638  
 std::numeric\_limits< unsigned char >, 2643  
 std::numeric\_limits< unsigned int >, 2648  
 std::numeric\_limits< unsigned long >, 2654  
 std::numeric\_limits< unsigned long long >, 2659  
 std::numeric\_limits< unsigned short >, 2664  
 std::numeric\_limits< wchar\_t >, 2669

## hash

std::\_\_cxx11::collate< \_CharT >, 1860

## Hash-Based, 141

## hash\_bytes.h, 3412

## hash\_eq\_fn.hpp, 3634

## hash\_exponential\_size\_policy

\_\_gnu\_pbds::hash\_exponential\_size\_policy< Size\_Type  
 >, 2223

## hash\_exponential\_size\_policy\_imp.hpp, 3646

## hash\_fun.h, 3360

## hash\_function

std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
 loc >, 3186  
 std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
 \_Alloc >, 3215  
 std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
 loc >, 3244  
 std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
 >, 3272

## hash\_load\_check\_resize\_trigger

\_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< Ex-  
 ternal\_Load\_Access, Size\_Type >, 2225

## hash\_load\_check\_resize\_trigger\_imp.hpp, 3646

## hash\_load\_check\_resize\_trigger\_size\_base.hpp, 3646

## hash\_map, 3360

## hash\_policy.hpp, 3653

## hash\_prime\_size\_policy

\_\_gnu\_pbds::hash\_prime\_size\_policy, 2232

## hash\_prime\_size\_policy\_imp.hpp, 3647

## hash\_set, 3361

## hash\_standard\_resize\_policy

\_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy  
 Trigger\_Policy, External\_Size\_Access, Size\_Type

>, 2236

## hash\_standard\_resize\_policy\_imp.hpp, 3647

## hasher

std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
 loc >, 3171  
 std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
 \_Alloc >, 3201  
 std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
 loc >, 3230  
 std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
 >, 3256

## Hashes, 315

## hashtable.h, 3413

## hashtable\_policy.h, 3414

## Heap, 86

is\_heap, 87

is\_heap\_until, 88

make\_heap, 89

pop\_heap, 89, 90

push\_heap, 90, 91

sort\_heap, 91

## Heap-Based, 142

## helper\_functions.h, 3578

## hermite

Mathematical Special Functions, 208

TR1 Mathematical Special Functions, 250

## hermitef

Mathematical Special Functions, 208

## hermitel

Mathematical Special Functions, 209

## hex

std, 599

std::basic\_fstream< \_CharT, \_Traits >, 1158

std::basic\_ifstream< \_CharT, \_Traits >, 1200

std::basic\_ios< \_CharT, \_Traits >, 1223

std::basic\_iostream< \_CharT, \_Traits >, 1270

std::basic\_istream< \_CharT, \_Traits >, 1307

std::basic\_istream< \_CharT, \_Traits >::sentry, 2895

std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >,  
 1346

std::basic\_ofstream< \_CharT, \_Traits >, 1380

std::basic\_ostream< \_CharT, \_Traits >, 1410

std::basic\_ostream< \_CharT, \_Traits >::sentry, 2923

std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >,  
 1444

std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >,  
 1715

std::ios\_base, 2268

## hexfloat

std, 599

## high\_resolution\_clock

Time, 365

## hypers

Time, 365

- hyperg
  - Mathematical Special Functions, 209
  - std::tr1, 725
- hypergf
  - Mathematical Special Functions, 209
  - std::tr1, 726
- hypergl
  - Mathematical Special Functions, 210
  - std::tr1, 726
- I/O, 163
  - filebuf, 165
  - fstream, 165
  - ifstream, 165
  - ios, 165
  - iostream, 165
  - istream, 165
  - istringstream, 165
  - ofstream, 165
  - ostream, 165
  - ostingstream, 165
  - streambuf, 165
  - stringbuf, 165
  - stringstream, 166
  - wfilebuf, 166
  - wfstream, 166
  - wifstream, 166
  - wios, 166
  - wiostream, 166
  - wistream, 166
  - wistringstream, 166
  - wofstream, 166
  - wostream, 166
  - wostingstream, 166
  - wstreambuf, 167
  - wstringbuf, 167
  - wstringstream, 167
- icase
  - std::regex\_constants, 717
- id
  - std::\_\_cxx11::collate< \_CharT >, 1861
  - std::ctype< \_CharT >, 1923
  - std::ctype< char >, 1958
  - std::ctype< wchar\_t >, 1976
  - std::ctype\_byname< \_CharT >, 1990
  - std::ctype\_byname< char >, 2013
  - std::locale::id, 2239
  - std::messages< \_CharT >, 2431
  - std::messages\_byname< \_CharT >, 2433
  - std::money\_get< \_CharT, \_InIter >, 2444
  - std::money\_put< \_CharT, \_OutIter >, 2448
  - std::moneypunct< \_CharT, \_Intl >, 2457
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2465
  - std::num\_get< \_CharT, \_InIter >, 2559
  - std::num\_put< \_CharT, \_OutIter >, 2572
  - std::numput< \_CharT >, 2677
  - std::numput\_byname< \_CharT >, 2681
  - std::time\_get< \_CharT, \_InIter >, 3072
  - std::time\_get\_byname< \_CharT, \_InIter >, 3082
  - std::time\_put< \_CharT, \_OutIter >, 3087
  - std::time\_put\_byname< \_CharT, \_OutIter >, 3090
- identity\_element
  - SGL, 154
- ifstream
  - I/O, 165
- ignore
  - std, 646
  - std::basic\_fstream< \_CharT, \_Traits >, 1128, 1129
  - std::basic\_ifstream< \_CharT, \_Traits >, 1179, 1180
  - std::basic\_iostream< \_CharT, \_Traits >, 1240, 1241
  - std::basic\_istream< \_CharT, \_Traits >, 1286
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2877, 2878
  - std::basic\_istringstream< \_CharT, \_Traits, \_Alloc >, 1325
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1685, 1686
- imbue
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2088
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2994
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3013
  - std::basic\_filebuf< \_CharT, \_Traits >, 1096
  - std::basic\_fstream< \_CharT, \_Traits >, 1129
  - std::basic\_ifstream< \_CharT, \_Traits >, 1180
  - std::basic\_ios< \_CharT, \_Traits >, 1213
  - std::basic\_iostream< \_CharT, \_Traits >, 1241
  - std::basic\_istream< \_CharT, \_Traits >, 1288
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2878
  - std::basic\_istringstream< \_CharT, \_Traits, \_Alloc >, 1326
  - std::basic\_ofstream< \_CharT, \_Traits >, 1360
  - std::basic\_ostream< \_CharT, \_Traits >, 1392
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2907
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1426
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1453
  - std::basic\_streambuf< \_CharT, \_Traits >, 1462
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1656
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1686
  - std::ios\_base, 2262
  - std::regex\_traits< \_Ch\_type >, 2824
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3333
- in
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, 744
  - std::basic\_fstream< \_CharT, \_Traits >, 1158

- std::basic\_ifstream< \_CharT, \_Traits >, 1200
- std::basic\_ios< \_CharT, \_Traits >, 1223
- std::basic\_iostream< \_CharT, \_Traits >, 1270
- std::basic\_istream< \_CharT, \_Traits >, 1307
- std::basic\_istream< \_CharT, \_Traits >::sentry, 2895
- std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1346
- std::basic\_ofstream< \_CharT, \_Traits >, 1380
- std::basic\_ostream< \_CharT, \_Traits >, 1410
- std::basic\_ostream< \_CharT, \_Traits >::sentry, 2923
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1444
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1715
- std::codecvt< \_InternT, \_ExternT, \_StateT >, 1803
- std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1809, 1810
- std::codecvt< char, char, mbstate\_t >, 1818, 1819
- std::codecvt< char16\_t, char, mbstate\_t >, 1827, 1828
- std::codecvt< char32\_t, char, mbstate\_t >, 1836, 1837
- std::codecvt< wchar\_t, char, mbstate\_t >, 1845, 1846
- std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1854
- std::ios\_base, 2268
- in\_avail
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2089
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2994
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3013
  - std::basic\_filebuf< \_CharT, \_Traits >, 1096
  - std::basic\_streambuf< \_CharT, \_Traits >, 1462
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1657
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3333
- in\_place
  - Optional values, 287
- includes
  - Set Operations, 93
- inclusive\_scan
  - Generalized Numeric operations, 9, 10
- increment
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2323
- independent\_bits\_engine
  - std::independent\_bits\_engine< \_\_RandomNumberEngine, \_\_w, \_UIntType >, 2242, 2243
- index\_sequence
  - std, 579
- index\_sequence\_for
  - std, 579
- indirect.h, 3414
- indirect\_array
  - Numeric Arrays, 219
- indirect\_array.h, 3414
- infinity
  - std::numeric\_limits< \_Tp >, 2573
  - std::numeric\_limits< bool >, 2579
  - std::numeric\_limits< char >, 2584
  - std::numeric\_limits< char16\_t >, 2589
  - std::numeric\_limits< char32\_t >, 2595
  - std::numeric\_limits< double >, 2600
  - std::numeric\_limits< float >, 2605
  - std::numeric\_limits< int >, 2610
  - std::numeric\_limits< long >, 2616
  - std::numeric\_limits< long double >, 2621
  - std::numeric\_limits< long long >, 2626
  - std::numeric\_limits< short >, 2631
  - std::numeric\_limits< signed char >, 2637
  - std::numeric\_limits< unsigned char >, 2642
  - std::numeric\_limits< unsigned int >, 2647
  - std::numeric\_limits< unsigned long >, 2652
  - std::numeric\_limits< unsigned long long >, 2658
  - std::numeric\_limits< unsigned short >, 2663
  - std::numeric\_limits< wchar\_t >, 2668
- info\_fn\_imps.hpp, 3621, 3622
- init
  - std::basic\_fstream< \_CharT, \_Traits >, 1130
  - std::basic\_ifstream< \_CharT, \_Traits >, 1181
  - std::basic\_ios< \_CharT, \_Traits >, 1214
  - std::basic\_iostream< \_CharT, \_Traits >, 1242
  - std::basic\_istream< \_CharT, \_Traits >, 1288
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2879
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1326
  - std::basic\_ofstream< \_CharT, \_Traits >, 1361
  - std::basic\_ostream< \_CharT, \_Traits >, 1393
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2907
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1426
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1687
- initializer\_list, 3804
- inner\_product
  - Generalized Numeric operations, 11
- inplace\_merge
  - Sorting, 63, 64
- inplace\_vector, 3688
- insert
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 836–840
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1519–1525
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1604–1611, 1613, 1614
  - std::deque< \_Tp, \_Alloc >, 2044–2046
  - std::list< \_Tp, \_Alloc >, 2337–2339

- `std::map< _Key, _Tp, _Compare, _Alloc >`, [2394–2397](#)
- `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [2489–2492](#)
- `std::multiset< _Key, _Compare, _Alloc >`, [2516–2518](#)
- `std::set< _Key, _Compare, _Alloc >`, [2944, 2945](#)
- `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3186–3189](#)
- `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3215, 3216, 3218](#)
- `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3244–3246](#)
- `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3272–3274](#)
- `std::vector< _Tp, _Alloc >`, [3300–3302](#)
- `std::vector< bool, _Alloc >`, [3321, 3322](#)
- `insert_after`
  - `std::forward_list< _Tp, _Alloc >`, [2151, 2152](#)
- `insert_fn_imps.hpp`, [3622, 3623](#)
- `insert_iterator`
  - `std::insert_iterator< _Container >`, [2253](#)
- `insert_join_fn_imps.hpp`, [3642](#)
- `insert_no_store_hash_fn_imps.hpp`, [3632](#)
- `insert_or_assign`
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, [2398](#)
- `insert_store_hash_fn_imps.hpp`, [3632](#)
- `inserter`
  - Iterators, [171](#)
- `int_type`
  - `std::basic_ios< _CharT, _Traits >`, [1207](#)
  - `std::basic_streambuf< _CharT, _Traits >`, [1459](#)
  - `std::istreambuf_iterator< _CharT, _Traits >`, [2294](#)
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, [3330](#)
- `internal`
  - `std`, [599](#)
  - `std::basic_fstream< _CharT, _Traits >`, [1158](#)
  - `std::basic_ifstream< _CharT, _Traits >`, [1200](#)
  - `std::basic_ios< _CharT, _Traits >`, [1223](#)
  - `std::basic_iostream< _CharT, _Traits >`, [1270](#)
  - `std::basic_istream< _CharT, _Traits >`, [1307](#)
  - `std::basic_istream< _CharT, _Traits >::sentry`, [2895](#)
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, [1346](#)
  - `std::basic_ofstream< _CharT, _Traits >`, [1380](#)
  - `std::basic_ostream< _CharT, _Traits >`, [1411](#)
  - `std::basic_ostream< _CharT, _Traits >::sentry`, [2923](#)
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, [1444](#)
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, [1715](#)
  - `std::ios_base`, [2268](#)
- `intervals`
  - `std::piecewise_constant_distribution< _RealType >`, [2743](#)
- `std::piecewise_linear_distribution< _RealType >`, [2747](#)
- `intl`
  - `std::moneypunct< _CharT, _Intl >`, [2457](#)
- Invalidation Guarantees, [145](#)
- `invoke.h`, [3414](#)
- `io_errc`
  - `std`, [581](#)
- `iomanip`, [3688](#)
- `ios`, [3690](#)
  - I/O, [165](#)
- `ios_base.h`, [3415](#)
- `iosfwd`, [3690](#)
- `iostate`
  - `std::basic_fstream< _CharT, _Traits >`, [1117](#)
  - `std::basic_ifstream< _CharT, _Traits >`, [1168](#)
  - `std::basic_ios< _CharT, _Traits >`, [1208](#)
  - `std::basic_iostream< _CharT, _Traits >`, [1231](#)
  - `std::basic_istream< _CharT, _Traits >`, [1277](#)
  - `std::basic_istream< _CharT, _Traits >::sentry`, [2868](#)
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, [1316](#)
  - `std::basic_ofstream< _CharT, _Traits >`, [1354](#)
  - `std::basic_ostream< _CharT, _Traits >`, [1387](#)
  - `std::basic_ostream< _CharT, _Traits >::sentry`, [2901](#)
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, [1419](#)
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, [1676](#)
  - `std::ios_base`, [2260](#)
- `iostream`, [3691](#)
  - I/O, [165](#)
- `iota`
  - `__gnu_cxx`, [394](#)
  - Generalized Numeric operations, [12](#)
- `is`
  - `std::__ctype_abstract_base< _CharT >`, [757](#)
  - `std::ctype< _CharT >`, [1917, 1918](#)
  - `std::ctype< char >`, [1942–1944](#)
  - `std::ctype< wchar_t >`, [1971, 1972](#)
  - `std::ctype_byname< _CharT >`, [1985](#)
  - `std::ctype_byname< char >`, [2003, 2004](#)
- `is_always_equal`
  - `__gnu_cxx::__alloc_traits< _Alloc, typename >`, [734](#)
  - `std::allocator_traits< _Alloc >`, [1033](#)
  - `std::allocator_traits< allocator< _Tp > >`, [1039](#)
  - `std::allocator_traits< allocator< void > >`, [1047, 1048](#)
  - `std::allocator_traits< pmr::polymorphic_allocator< _Tp > >`, [1056](#)
- `is_bind_expression_v`
  - `std::experimental`, [703](#)
- `is_bounded`



- std::\_\_numeric\_limits\_base, 789
- std::numeric\_limits< \_Tp >, 2575
- std::numeric\_limits< bool >, 2580
- std::numeric\_limits< char >, 2586
- std::numeric\_limits< char16\_t >, 2591
- std::numeric\_limits< char32\_t >, 2596
- std::numeric\_limits< double >, 2601
- std::numeric\_limits< float >, 2607
- std::numeric\_limits< int >, 2612
- std::numeric\_limits< long >, 2617
- std::numeric\_limits< long double >, 2622
- std::numeric\_limits< long long >, 2628
- std::numeric\_limits< short >, 2633
- std::numeric\_limits< signed char >, 2638
- std::numeric\_limits< unsigned char >, 2643
- std::numeric\_limits< unsigned int >, 2649
- std::numeric\_limits< unsigned long >, 2654
- std::numeric\_limits< unsigned long long >, 2659
- std::numeric\_limits< unsigned short >, 2664
- std::numeric\_limits< wchar\_t >, 2670
- is\_corresponding\_member
  - Metaprogramming, 352
- is\_detected
  - Detection idiom, 285
- is\_detected\_convertible
  - Detection idiom, 285
- is\_detected\_convertible\_v
  - Detection idiom, 285
- is\_detected\_exact
  - Detection idiom, 285
- is\_detected\_exact\_v
  - Detection idiom, 286
- is\_detected\_v
  - Detection idiom, 286
- is\_emptyset
  - std::tr2::bool\_set, 1771
- is\_exact
  - std::\_\_numeric\_limits\_base, 789
  - std::numeric\_limits< \_Tp >, 2575
  - std::numeric\_limits< bool >, 2580
  - std::numeric\_limits< char >, 2586
  - std::numeric\_limits< char16\_t >, 2591
  - std::numeric\_limits< char32\_t >, 2596
  - std::numeric\_limits< double >, 2601
  - std::numeric\_limits< float >, 2607
  - std::numeric\_limits< int >, 2612
  - std::numeric\_limits< long >, 2617
  - std::numeric\_limits< long double >, 2622
  - std::numeric\_limits< long long >, 2628
  - std::numeric\_limits< short >, 2633
  - std::numeric\_limits< signed char >, 2638
  - std::numeric\_limits< unsigned char >, 2643
  - std::numeric\_limits< unsigned int >, 2649
  - std::numeric\_limits< unsigned long >, 2654
- std::numeric\_limits< unsigned long long >, 2659
- std::numeric\_limits< unsigned short >, 2664
- std::numeric\_limits< wchar\_t >, 2670
- is\_grow\_needed
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1777
  - \_\_gnu\_pbds::sample\_resize\_trigger, 2851
- is\_heap
  - Heap, 87
- is\_heap\_until
  - Heap, 88
- is\_iec559
  - std::\_\_numeric\_limits\_base, 789
  - std::numeric\_limits< \_Tp >, 2575
  - std::numeric\_limits< bool >, 2581
  - std::numeric\_limits< char >, 2586
  - std::numeric\_limits< char16\_t >, 2591
  - std::numeric\_limits< char32\_t >, 2596
  - std::numeric\_limits< double >, 2602
  - std::numeric\_limits< float >, 2607
  - std::numeric\_limits< int >, 2612
  - std::numeric\_limits< long >, 2617
  - std::numeric\_limits< long double >, 2623
  - std::numeric\_limits< long long >, 2628
  - std::numeric\_limits< short >, 2633
  - std::numeric\_limits< signed char >, 2638
  - std::numeric\_limits< unsigned char >, 2644
  - std::numeric\_limits< unsigned int >, 2649
  - std::numeric\_limits< unsigned long >, 2654
  - std::numeric\_limits< unsigned long long >, 2659
  - std::numeric\_limits< unsigned short >, 2665
  - std::numeric\_limits< wchar\_t >, 2670
- is\_indeterminate
  - std::tr2::bool\_set, 1771
- is\_integer
  - std::\_\_numeric\_limits\_base, 789
  - std::numeric\_limits< \_Tp >, 2575
  - std::numeric\_limits< bool >, 2581
  - std::numeric\_limits< char >, 2586
  - std::numeric\_limits< char16\_t >, 2591
  - std::numeric\_limits< char32\_t >, 2596
  - std::numeric\_limits< double >, 2602
  - std::numeric\_limits< float >, 2607
  - std::numeric\_limits< int >, 2612
  - std::numeric\_limits< long >, 2617
  - std::numeric\_limits< long double >, 2623
  - std::numeric\_limits< long long >, 2628
  - std::numeric\_limits< short >, 2633
  - std::numeric\_limits< signed char >, 2638
  - std::numeric\_limits< unsigned char >, 2644
  - std::numeric\_limits< unsigned int >, 2649
  - std::numeric\_limits< unsigned long >, 2654
  - std::numeric\_limits< unsigned long long >, 2659
  - std::numeric\_limits< unsigned short >, 2665

- std::numeric\_limits< wchar\_t >, 2670
- is\_layout\_compatible\_v
  - Metaprogramming, 353
- is\_modulo
  - std::\_\_numeric\_limits\_base, 789
  - std::numeric\_limits< \_Tp >, 2576
  - std::numeric\_limits< bool >, 2581
  - std::numeric\_limits< char >, 2586
  - std::numeric\_limits< char16\_t >, 2591
  - std::numeric\_limits< char32\_t >, 2597
  - std::numeric\_limits< double >, 2602
  - std::numeric\_limits< float >, 2607
  - std::numeric\_limits< int >, 2612
  - std::numeric\_limits< long >, 2617
  - std::numeric\_limits< long double >, 2623
  - std::numeric\_limits< long long >, 2628
  - std::numeric\_limits< short >, 2633
  - std::numeric\_limits< signed char >, 2639
  - std::numeric\_limits< unsigned char >, 2644
  - std::numeric\_limits< unsigned int >, 2649
  - std::numeric\_limits< unsigned long >, 2654
  - std::numeric\_limits< unsigned long long >, 2660
  - std::numeric\_limits< unsigned short >, 2665
  - std::numeric\_limits< wchar\_t >, 2670
- is\_open
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2089
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2994
  - std::basic\_filebuf< \_CharT, \_Traits >, 1096
  - std::basic\_fstream< \_CharT, \_Traits >, 1130
  - std::basic\_ifstream< \_CharT, \_Traits >, 1181
  - std::basic\_ofstream< \_CharT, \_Traits >, 1361
- is\_partitioned
  - Mutating, 25
- is\_permutation
  - Non-Mutating, 53, 54
- is\_placeholder\_v
  - std::experimental, 703
- is\_pointer\_interconvertible\_base\_of\_v
  - Metaprogramming, 354
- is\_pointer\_interconvertible\_with\_class
  - Metaprogramming, 353
- is\_resize\_needed
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger
    - External\_Load\_Access, Size\_Type >, 1777
  - \_\_gnu\_pbds::sample\_resize\_policy, 2848
  - \_\_gnu\_pbds::sample\_resize\_trigger, 2851
- is\_signed
  - std::\_\_numeric\_limits\_base, 789
  - std::numeric\_limits< \_Tp >, 2576
  - std::numeric\_limits< bool >, 2581
  - std::numeric\_limits< char >, 2586
  - std::numeric\_limits< char16\_t >, 2591
  - std::numeric\_limits< char32\_t >, 2597
  - std::numeric\_limits< double >, 2602
  - std::numeric\_limits< float >, 2607
  - std::numeric\_limits< int >, 2612
  - std::numeric\_limits< long >, 2617
  - std::numeric\_limits< long double >, 2623
  - std::numeric\_limits< long long >, 2628
  - std::numeric\_limits< short >, 2633
  - std::numeric\_limits< signed char >, 2639
  - std::numeric\_limits< unsigned char >, 2644
  - std::numeric\_limits< unsigned int >, 2649
  - std::numeric\_limits< unsigned long >, 2654
  - std::numeric\_limits< unsigned long long >, 2660
  - std::numeric\_limits< unsigned short >, 2665
  - std::numeric\_limits< wchar\_t >, 2670
- is\_singleton
  - std::tr2::bool\_set, 1771
- is\_sorted
  - Sorting, 64, 65
- is\_sorted\_until
  - Sorting, 65
- is\_specialized
  - std::\_\_numeric\_limits\_base, 789
  - std::numeric\_limits< \_Tp >, 2576
  - std::numeric\_limits< bool >, 2581
  - std::numeric\_limits< char >, 2586
  - std::numeric\_limits< char16\_t >, 2592
  - std::numeric\_limits< char32\_t >, 2597
  - std::numeric\_limits< double >, 2602
  - std::numeric\_limits< float >, 2607
  - std::numeric\_limits< int >, 2612
  - std::numeric\_limits< long >, 2618
  - std::numeric\_limits< long double >, 2623
  - std::numeric\_limits< long long >, 2628
  - std::numeric\_limits< short >, 2634
  - std::numeric\_limits< signed char >, 2639
  - std::numeric\_limits< unsigned char >, 2644
  - std::numeric\_limits< unsigned int >, 2649
  - std::numeric\_limits< unsigned long >, 2655
  - std::numeric\_limits< unsigned long long >, 2660
  - std::numeric\_limits< unsigned short >, 2665
  - std::numeric\_limits< wchar\_t >, 2670
- isalnum
  - std, 599
- isalpha
  - std, 599
- isblank
  - std, 600
- iscntrl
  - std, 600
- isctype
  - std::regex\_traits< \_Ch\_type >, 2824
- isdigit
  - std, 600
- isgraph
  - std, 600



islower  
     std, 600  
 isprint  
     std, 600  
 ispunct  
     std, 601  
 isspace  
     std, 601  
 istream, 3692  
     I/O, 165  
 istream.tcc, 3416  
 istream\_iterator  
     std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2292  
 istream\_type  
     std::istreambuf\_iterator< \_CharT, \_Traits >, 2294  
 istreambuf\_iterator  
     std::istreambuf\_iterator< \_CharT, \_Traits >, 2295, 2296  
 istringstream  
     I/O, 165  
 isupper  
     std, 601  
 isxdigit  
     std, 601  
 iter\_swap  
     Mutating, 25  
 iter\_type  
     std::money\_get< \_CharT, \_InIter >, 2441  
     std::money\_put< \_CharT, \_OutIter >, 2445  
     std::num\_get< \_CharT, \_InIter >, 2546  
     std::num\_put< \_CharT, \_OutIter >, 2561  
     std::time\_get< \_CharT, \_InIter >, 3063  
     std::time\_put< \_CharT, \_OutIter >, 3085  
 iterator, 3693  
     std::set< \_Key, \_Compare, \_Alloc >, 2932  
     std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3171  
     std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3201  
     std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3230  
     std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3256  
 Iterator Tags, 173  
 iterator.h, 3731  
 iterator.hpp, 3652  
 iterator\_category  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator< Node, Const\_Iterator, Iterator, \_Alloc >, 1724  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator< Node, Const\_Iterator, Iterator, \_Alloc >, 1728  
     \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1738  
     \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1742  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, 2306  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, 2310  
     std::back\_insert\_iterator< \_Container >, 1077  
     std::front\_insert\_iterator< \_Container >, 2163  
     std::insert\_iterator< \_Container >, 2253  
     std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2292  
     std::istreambuf\_iterator< \_CharT, \_Traits >, 2295  
     std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 2300  
     std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 2687  
     std::ostreambuf\_iterator< \_CharT, \_Traits >, 2690  
     std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 2795  
 iterator\_concepts.h, 3417  
 iterator\_fn\_imps.hpp, 3635  
 Iterators, 167  
     \_\_iterator\_category, 170  
     back\_inserter, 171  
     front\_inserter, 171  
     inserter, 171  
     make\_reverse\_iterator, 172  
     operator==, 172  
 iterators\_fn\_imps.hpp, 3623, 3624  
 iword  
     std::basic\_fstream< \_CharT, \_Traits >, 1130  
     std::basic\_ifstream< \_CharT, \_Traits >, 1181  
     std::basic\_ios< \_CharT, \_Traits >, 1214  
     std::basic\_iostream< \_CharT, \_Traits >, 1242  
     std::basic\_istream< \_CharT, \_Traits >, 1288  
     std::basic\_istream< \_CharT, \_Traits >::sentry, 2879  
     std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1326  
     std::basic\_ofstream< \_CharT, \_Traits >, 1361  
     std::basic\_ostream< \_CharT, \_Traits >, 1393  
     std::basic\_ostream< \_CharT, \_Traits >::sentry, 2907  
     std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1426  
     std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1687  
     std::ios\_base, 2262  
  
 k  
     std::negative\_binomial\_distribution< \_IntType >, 2530  
 key\_comp  
     std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2399  
     std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2493  
     std::multiset< \_Key, \_Compare, \_Alloc >, 2518

- std::set< \_Key, \_Compare, \_Alloc >, [2945](#)
- key\_compare
  - std::set< \_Key, \_Compare, \_Alloc >, [2932](#)
- key\_eq
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3189](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3219](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3246](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3274](#)
- key\_equal
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3171](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3201](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3230](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3256](#)
- key\_type
  - std::set< \_Key, \_Compare, \_Alloc >, [2932](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3171](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3202](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3230](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3256](#)
- kill\_dependency
  - Atomics, [110](#)
- L1\_cache\_size
  - \_\_gnu\_parallel::Settings, [1012](#)
- L2\_cache\_size
  - \_\_gnu\_parallel::Settings, [1012](#)
- laguerre
  - Mathematical Special Functions, [210](#)
  - TR1 Mathematical Special Functions, [250](#)
- laguerref
  - Mathematical Special Functions, [210](#)
- laguerrel
  - Mathematical Special Functions, [211](#)
- lambda
  - std::exponential\_distribution< \_RealType >, [2122](#)
- latch, [3694](#)
- launch
  - Futures, [114](#)
- lcm
  - std::experimental, [701](#)
- left
  - std, [601](#)
- std::basic\_fstream< \_CharT, \_Traits >, [1158](#)
- std::basic\_ifstream< \_CharT, \_Traits >, [1201](#)
- std::basic\_ios< \_CharT, \_Traits >, [1224](#)
- std::basic\_iostream< \_CharT, \_Traits >, [1270](#)
- std::basic\_istream< \_CharT, \_Traits >, [1307](#)
- std::basic\_istream< \_CharT, \_Traits >::sentry, [2895](#)
- std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1347](#)
- std::basic\_ofstream< \_CharT, \_Traits >, [1380](#)
- std::basic\_ostream< \_CharT, \_Traits >, [1411](#)
- std::basic\_ostream< \_CharT, \_Traits >::sentry, [2923](#)
- std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1444](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1715](#)
- std::ios\_base, [2269](#)
- left\_child\_next\_sibling\_heap.hpp, [3639](#)
- left\_child\_next\_sibling\_heap\_const\_iterator
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, [2306](#)
- left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, [2310](#)
- legendre
  - Mathematical Special Functions, [211](#)
  - TR1 Mathematical Special Functions, [251](#)
- legendref
  - Mathematical Special Functions, [211](#)
- legendrel
  - Mathematical Special Functions, [212](#)
- length
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [841](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1526](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1614](#), [1615](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2413](#)
  - std::regex\_traits< \_Ch\_type >, [2825](#)
  - std::sub\_match< \_Biter >, [3036](#)
- lexicographical\_compare
  - Sorting, [66](#)
- lexicographical\_compare\_3way
  - SGI, [155](#)
- lexicographical\_compare\_three\_way
  - Sorting, [67](#)
- lfts\_config.h, [3602](#)
- Library Fundamentals TS, [280](#)
- limits, [3694](#)
- linear\_congruential\_engine
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [2320](#)
- linear\_probe\_fn\_imp.hpp, [3636](#)
- list, [3695](#), [3696](#)

- std::list< \_Tp, \_Alloc >, 2331, 2332
- List-Based, 143
- list.tcc, 3417
- list\_partition
  - \_\_gnu\_parallel, 452
- list\_partition.h, 3732
- list\_update
  - \_\_gnu\_pbds::list\_update< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >, 2348
- list\_update\_policy.hpp, 3654
- load\_factor
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3189
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3219
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3246
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3274
- local\_iterator
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3171
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3202
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3230
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3256
- local\_time\_format
  - Time, 369
- locale, 3697
  - std::locale, 2350–2352
- locale\_classes.h, 3418
- locale\_classes.tcc, 3418
- locale\_conv.h, 3418
- locale\_facets.h, 3419
- locale\_facets.tcc, 3421
- locale\_facets\_nonio.h, 3421
- locale\_facets\_nonio.tcc, 3422
- localefwd.h, 3423
- Locales, 173
  - has\_facet, 174
  - use\_facet, 175
- lock
  - Mutexes, 117
- log
  - Complex Numbers, 181
  - math.h, 3701
- log10
  - Complex Numbers, 182
  - math.h, 3701
- logic\_error
  - std::logic\_error, 2358
- Logical operator traits, 286
- lookup\_classname
  - std::regex\_traits< \_Ch\_type >, 2825
- lookup\_collatename
  - std::regex\_traits< \_Ch\_type >, 2826
- losertree.h, 3732
- lower\_bound
  - Binary Search, 84
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2399, 2400
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2493, 2494
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2518, 2519
  - std::set< \_Key, \_Compare, \_Alloc >, 2945, 2946
- lowest
  - std::numeric\_limits< \_Tp >, 2573
  - std::numeric\_limits< bool >, 2579
  - std::numeric\_limits< char >, 2584
  - std::numeric\_limits< char16\_t >, 2590
  - std::numeric\_limits< char32\_t >, 2595
  - std::numeric\_limits< double >, 2600
  - std::numeric\_limits< float >, 2605
  - std::numeric\_limits< int >, 2611
  - std::numeric\_limits< long >, 2616
  - std::numeric\_limits< long double >, 2621
  - std::numeric\_limits< long long >, 2626
  - std::numeric\_limits< short >, 2632
  - std::numeric\_limits< signed char >, 2637
  - std::numeric\_limits< unsigned char >, 2642
  - std::numeric\_limits< unsigned int >, 2647
  - std::numeric\_limits< unsigned long >, 2653
  - std::numeric\_limits< unsigned long long >, 2658
  - std::numeric\_limits< unsigned short >, 2663
  - std::numeric\_limits< wchar\_t >, 2668
- lu\_counter\_metadata.hpp, 3641
- lu\_map\_.hpp, 3640
- macros.h, 3580
  - \_GLIBCXX\_DEBUG\_VERIFY\_AT\_F, 3583
  - \_glibcxx\_check\_erase, 3581
  - \_glibcxx\_check\_erase\_after, 3581
  - \_glibcxx\_check\_erase\_range, 3581
  - \_glibcxx\_check\_erase\_range\_after, 3581
  - \_glibcxx\_check\_heap\_pred, 3581
  - \_glibcxx\_check\_insert, 3581
  - \_glibcxx\_check\_insert\_after, 3581
  - \_glibcxx\_check\_insert\_range, 3582
  - \_glibcxx\_check\_insert\_range\_after, 3582
  - \_glibcxx\_check\_partitioned\_lower, 3582
  - \_glibcxx\_check\_partitioned\_lower\_pred, 3582
  - \_glibcxx\_check\_partitioned\_upper\_pred, 3582
  - \_glibcxx\_check\_sorted\_pred, 3583
- make\_array
  - Array creation functions, 282
- make\_boyer\_moore\_horspool\_searcher

- std::experimental, 701
- make\_boyer\_moore\_searcher
  - std::experimental, 702
- make\_default\_searcher
  - std::experimental, 702
- make\_error\_code
  - Diagnostics, 124
  - Futures, 115
- make\_error\_condition
  - Diagnostics, 124
  - Futures, 115
- make\_exception\_ptr
  - Exceptions, 130
- make\_heap
  - Heap, 89
- make\_index\_sequence
  - std, 579
- make\_integer\_sequence
  - std, 579
- make\_ostream\_joiner
  - std::experimental, 702
- make\_pair
  - Utilities, 301
- make\_reverse\_iterator
  - Iterators, 172
- make\_shared
  - Pointer Abstractions, 338
- make\_signed\_t
  - Metaprogramming, 351
- make\_tuple
  - Utilities, 302
- make\_unsigned\_t
  - Metaprogramming, 351
- malloc\_allocator.h, 3609
- map, 3697, 3698
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2382–2385
- map.h, 3583
- mapped\_type
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3171
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3202
- mark\_count
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1453
- mask\_array
  - Numeric Arrays, 219
- mask\_array.h, 3423
- mask\_based\_range\_hashing.hpp, 3636
- match\_any
  - std::regex\_constants, 717
- match\_continuous
  - std::regex\_constants, 717
- match\_default
  - std::regex\_constants, 717
- match\_flag\_type
  - std::regex\_constants, 710
- match\_not\_bol
  - std::regex\_constants, 717
- match\_not\_bow
  - std::regex\_constants, 717
- match\_not\_eol
  - std::regex\_constants, 717
- match\_not\_eow
  - std::regex\_constants, 718
- match\_not\_null
  - std::regex\_constants, 718
- match\_prev\_avail
  - std::regex\_constants, 718
- match\_results
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2410, 2411
- math.h, 3699
  - abs, 3700
  - cos, 3700
  - cosh, 3701
  - exp, 3701
  - fabs, 3701
  - log, 3701
  - log10, 3701
  - pow, 3701
  - sin, 3701
  - sinh, 3701
  - sqrt, 3702
  - tan, 3702
  - tanh, 3702
- Mathematical Special Functions, 188
  - airy\_ai, 192
  - airy\_aif, 192
  - airy\_ail, 192
  - airy\_bi, 192
  - airy\_bif, 192
  - airy\_bil, 192
  - assoc\_laguerre, 193
  - assoc\_laguerref, 193
  - assoc\_laguerrel, 194
  - assoc\_legendre, 194
  - assoc\_legendref, 195
  - assoc\_legendrel, 195
  - beta, 195
  - betaf, 196
  - betal, 196
  - comp\_ellint\_1, 196
  - comp\_ellint\_1f, 197
  - comp\_ellint\_1l, 197
  - comp\_ellint\_2, 197
  - comp\_ellint\_2f, 198
  - comp\_ellint\_2l, 198
  - comp\_ellint\_3, 198

comp\_ellint\_3f, 199  
 comp\_ellint\_3l, 199  
 conf\_hyperg, 199  
 conf\_hypergf, 199  
 conf\_hypergl, 200  
 cyl\_bessel\_i, 200  
 cyl\_bessel\_if, 200  
 cyl\_bessel\_il, 201  
 cyl\_bessel\_j, 201  
 cyl\_bessel\_jf, 201  
 cyl\_bessel\_jl, 202  
 cyl\_bessel\_k, 202  
 cyl\_bessel\_kf, 202  
 cyl\_bessel\_kl, 203  
 cyl\_neumann, 203  
 cyl\_neumannf, 203  
 cyl\_neumannl, 204  
 ellint\_1, 204  
 ellint\_1f, 204  
 ellint\_1l, 205  
 ellint\_2, 205  
 ellint\_2f, 206  
 ellint\_2l, 206  
 ellint\_3, 206  
 ellint\_3f, 207  
 ellint\_3l, 207  
 expint, 207  
 expintf, 208  
 expintl, 208  
 hermite, 208  
 hermitef, 208  
 hermitel, 209  
 hyperg, 209  
 hypergf, 209  
 hypergl, 210  
 laguerre, 210  
 laguerref, 210  
 laguerrel, 211  
 legendre, 211  
 legendref, 211  
 legendrel, 212  
 riemann\_zeta, 212  
 riemann\_zetaf, 212  
 riemann\_zetal, 213  
 sph\_bessel, 213  
 sph\_besself, 213  
 sph\_bessell, 214  
 sph\_legendre, 214  
 sph\_legendref, 214  
 sph\_legendrel, 215  
 sph\_neumann, 215  
 sph\_neumannf, 215  
 sph\_neumannl, 215

max

\_\_gnu\_parallel, 452  
 Numeric Arrays, 223  
 Sorting, 67, 68  
 std::bernoulli\_distribution, 1718  
 std::binomial\_distribution< \_IntType >, 1749  
 std::cauchy\_distribution< \_RealType >, 1775  
 std::chi\_squared\_distribution< \_RealType >, 1798  
 std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2059  
 std::discrete\_distribution< \_IntType >, 2062  
 std::exponential\_distribution< \_RealType >, 2122  
 std::extreme\_value\_distribution< \_RealType >, 2125  
 std::fisher\_f\_distribution< \_RealType >, 2134  
 std::gamma\_distribution< \_RealType >, 2181  
 std::geometric\_distribution< \_IntType >, 2184  
 std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2243  
 std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2321  
 std::lognormal\_distribution< \_RealType >, 2366  
 std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 2426  
 std::negative\_binomial\_distribution< \_IntType >, 2530  
 std::normal\_distribution< \_RealType >, 2536  
 std::numeric\_limits< \_Tp >, 2574  
 std::numeric\_limits< bool >, 2579  
 std::numeric\_limits< char >, 2584  
 std::numeric\_limits< char16\_t >, 2590  
 std::numeric\_limits< char32\_t >, 2595  
 std::numeric\_limits< double >, 2600  
 std::numeric\_limits< float >, 2605  
 std::numeric\_limits< int >, 2611  
 std::numeric\_limits< long >, 2616  
 std::numeric\_limits< long double >, 2621  
 std::numeric\_limits< long long >, 2626  
 std::numeric\_limits< short >, 2632  
 std::numeric\_limits< signed char >, 2637  
 std::numeric\_limits< unsigned char >, 2642  
 std::numeric\_limits< unsigned int >, 2647  
 std::numeric\_limits< unsigned long >, 2653  
 std::numeric\_limits< unsigned long long >, 2658  
 std::numeric\_limits< unsigned short >, 2663  
 std::numeric\_limits< wchar\_t >, 2668  
 std::piecewise\_constant\_distribution< \_RealType >, 2743  
 std::piecewise\_linear\_distribution< \_RealType >, 2747  
 std::poisson\_distribution< \_IntType >, 2759  
 std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 2970  
 std::student\_t\_distribution< \_RealType >, 3031

- std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3040
- std::uniform\_int\_distribution< \_IntType >, 3143
- std::uniform\_real\_distribution< \_RealType >, 3146
- std::weibull\_distribution< \_RealType >, 3346
- max\_blocks\_per\_chunk
  - std::pmr::pool\_options, 2763
- max\_bucket\_count
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3190
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3219
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3246
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3275
- max\_count
  - \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, 2370
- max\_digits10
  - std::\_\_numeric\_limits\_base, 789
  - std::numeric\_limits< \_Tp >, 2576
  - std::numeric\_limits< bool >, 2581
  - std::numeric\_limits< char >, 2586
  - std::numeric\_limits< char16\_t >, 2592
  - std::numeric\_limits< char32\_t >, 2597
  - std::numeric\_limits< double >, 2602
  - std::numeric\_limits< float >, 2607
  - std::numeric\_limits< int >, 2613
  - std::numeric\_limits< long >, 2618
  - std::numeric\_limits< long double >, 2623
  - std::numeric\_limits< long long >, 2628
  - std::numeric\_limits< short >, 2634
  - std::numeric\_limits< signed char >, 2639
  - std::numeric\_limits< unsigned char >, 2644
  - std::numeric\_limits< unsigned int >, 2649
  - std::numeric\_limits< unsigned long >, 2655
  - std::numeric\_limits< unsigned long long >, 2660
  - std::numeric\_limits< unsigned short >, 2665
  - std::numeric\_limits< wchar\_t >, 2670
- max\_element
  - Sorting, 68, 69
- max\_element\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1012
- max\_exponent
  - std::\_\_numeric\_limits\_base, 790
  - std::numeric\_limits< \_Tp >, 2576
  - std::numeric\_limits< bool >, 2581
  - std::numeric\_limits< char >, 2586
  - std::numeric\_limits< char16\_t >, 2592
  - std::numeric\_limits< char32\_t >, 2597
  - std::numeric\_limits< double >, 2602
  - std::numeric\_limits< float >, 2607
  - std::numeric\_limits< int >, 2613
  - std::numeric\_limits< long >, 2618
  - std::numeric\_limits< long double >, 2623
  - std::numeric\_limits< long long >, 2628
  - std::numeric\_limits< short >, 2634
  - std::numeric\_limits< signed char >, 2639
  - std::numeric\_limits< unsigned char >, 2644
  - std::numeric\_limits< unsigned int >, 2649
  - std::numeric\_limits< unsigned long >, 2655
  - std::numeric\_limits< unsigned long long >, 2660
  - std::numeric\_limits< unsigned short >, 2665
  - std::numeric\_limits< wchar\_t >, 2670
- std::numeric\_limits< long >, 2618
- std::numeric\_limits< long double >, 2623
- std::numeric\_limits< long long >, 2628
- std::numeric\_limits< short >, 2634
- std::numeric\_limits< signed char >, 2639
- std::numeric\_limits< unsigned char >, 2644
- std::numeric\_limits< unsigned int >, 2649
- std::numeric\_limits< unsigned long >, 2655
- std::numeric\_limits< unsigned long long >, 2660
- std::numeric\_limits< unsigned short >, 2665
- std::numeric\_limits< wchar\_t >, 2670
- max\_exponent10
  - std::\_\_numeric\_limits\_base, 790
  - std::numeric\_limits< \_Tp >, 2576
  - std::numeric\_limits< bool >, 2581
  - std::numeric\_limits< char >, 2586
  - std::numeric\_limits< char16\_t >, 2592
  - std::numeric\_limits< char32\_t >, 2597
  - std::numeric\_limits< double >, 2602
  - std::numeric\_limits< float >, 2607
  - std::numeric\_limits< int >, 2613
  - std::numeric\_limits< long >, 2618
  - std::numeric\_limits< long double >, 2623
  - std::numeric\_limits< long long >, 2628
  - std::numeric\_limits< short >, 2634
  - std::numeric\_limits< signed char >, 2639
  - std::numeric\_limits< unsigned char >, 2644
  - std::numeric\_limits< unsigned int >, 2649
  - std::numeric\_limits< unsigned long >, 2655
  - std::numeric\_limits< unsigned long long >, 2660
  - std::numeric\_limits< unsigned short >, 2665
  - std::numeric\_limits< wchar\_t >, 2670
- max\_load\_factor
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3190
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3219
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3246, 3247
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3275
- max\_size
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 738
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 841
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1526
  - std::allocator\_traits< \_Alloc >, 1036
  - std::allocator\_traits< allocator< \_Tp > >, 1044
  - std::allocator\_traits< allocator< void > >, 1053
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, 1061, 1062
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1615
  - std::deque< \_Tp, \_Alloc >, 2046

- std::forward\_list< \_Tp, \_Alloc >, 2152
- std::list< \_Tp, \_Alloc >, 2339
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2400
- std::match\_results< \_Bi\_iter, \_Alloc >, 2413
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2494
- std::multiset< \_Key, \_Compare, \_Alloc >, 2520
- std::set< \_Key, \_Compare, \_Alloc >, 2948
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2076
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3190
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3221
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3247
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3275
- std::vector< \_Tp, \_Alloc >, 3302
- std::vector< bool, \_Alloc >, 3322
- max\_size\_type.h, 3424
- mdspan, 3702
- mean
  - std::normal\_distribution< \_RealType >, 2536
  - std::poisson\_distribution< \_IntType >, 2759
- mem\_fn
  - Function Objects, 308
- Memory, 317
  - align, 318
  - assume\_aligned, 319
  - uninitialized\_copy, 319
  - uninitialized\_copy\_n, 319
  - uninitialized\_default\_construct, 320
  - uninitialized\_default\_construct\_n, 320
  - uninitialized\_fill, 320
  - uninitialized\_fill\_n, 321
  - uninitialized\_move, 321
  - uninitialized\_move\_n, 322
  - uninitialized\_value\_construct, 322
  - uninitialized\_value\_construct\_n, 322
- memory, 3702–3704
- memory\_order
  - Atomics, 110
- memory\_resource, 3705, 3706
  - get\_default\_resource, 3706
  - null\_memory\_resource, 3706
  - set\_default\_resource, 3706
- memory\_resource.h, 3424
- memoryfwd.h, 3424
- merge
  - Sorting, 69, 70
  - std::forward\_list< \_Tp, \_Alloc >, 2153
  - std::list< \_Tp, \_Alloc >, 2339
- merge.h, 3732
- merge\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1012
- merge\_oversampling
  - \_\_gnu\_parallel::Settings, 1012
- mersenne\_twister\_engine
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 2426
- message
  - std::error\_category, 2112
  - std::error\_code, 2114
  - std::error\_condition, 2116
- messages
  - std::locale, 2356
  - std::messages< \_CharT >, 2430
- messages\_members.h, 3799
- metadata\_const\_reference
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1724
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1728
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 935
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 938
- metadata\_reference
  - \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, 2369
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, 2374
- metadata\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1724
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1729
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 935
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 938
  - \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, 2370
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, 2374
  - \_\_gnu\_pbds::sample\_update\_policy, 2856
- Metaprogramming, 346
  - add\_lvalue\_reference\_t, 350
  - add\_pointer\_t, 350
  - add\_rvalue\_reference\_t, 350
  - aligned\_storage\_t, 350
  - alignment\_value, 353
  - common\_reference\_t, 351



- common\_type\_t, [351](#)
- conditional\_t, [351](#)
- decay\_t, [351](#)
- enable\_if\_t, [351](#)
- false\_type, [351](#)
- is\_corresponding\_member, [352](#)
- is\_layout\_compatible\_v, [353](#)
- is\_pointer\_interconvertible\_base\_of\_v, [354](#)
- is\_pointer\_interconvertible\_with\_class, [353](#)
- make\_signed\_t, [351](#)
- make\_unsigned\_t, [351](#)
- remove\_all\_extents\_t, [351](#)
- remove\_extent\_t, [352](#)
- remove\_pointer\_t, [352](#)
- remove\_reference\_t, [352](#)
- result\_of\_t, [352](#)
- swap, [353](#)
- true\_type, [352](#)
- type, [352](#)
- underlying\_type\_t, [352](#)
- microseconds
  - Time, [365](#)
- milliseconds
  - Time, [365](#)
- min
  - \_\_gnu\_parallel, [452](#)
  - Numeric Arrays, [223](#)
  - Sorting, [70](#), [71](#)
  - std::bernoulli\_distribution, [1718](#)
  - std::binomial\_distribution< \_IntType >, [1749](#)
  - std::cauchy\_distribution< \_RealType >, [1775](#)
  - std::chi\_squared\_distribution< \_RealType >, [1798](#)
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2059](#)
  - std::discrete\_distribution< \_IntType >, [2062](#)
  - std::exponential\_distribution< \_RealType >, [2122](#)
  - std::extreme\_value\_distribution< \_RealType >, [2125](#)
  - std::fisher\_f\_distribution< \_RealType >, [2134](#)
  - std::gamma\_distribution< \_RealType >, [2181](#)
  - std::geometric\_distribution< \_IntType >, [2184](#)
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, [2244](#)
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [2321](#)
  - std::lognormal\_distribution< \_RealType >, [2366](#)
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [2426](#)
  - std::negative\_binomial\_distribution< \_IntType >, [2530](#)
  - std::normal\_distribution< \_RealType >, [2537](#)
  - std::numeric\_limits< \_Tp >, [2574](#)
  - std::numeric\_limits< bool >, [2579](#)
  - std::numeric\_limits< char >, [2584](#)
  - std::numeric\_limits< char16\_t >, [2590](#)
  - std::numeric\_limits< char32\_t >, [2595](#)
  - std::numeric\_limits< double >, [2600](#)
  - std::numeric\_limits< float >, [2605](#)
  - std::numeric\_limits< int >, [2611](#)
  - std::numeric\_limits< long >, [2616](#)
  - std::numeric\_limits< long double >, [2621](#)
  - std::numeric\_limits< long long >, [2626](#)
  - std::numeric\_limits< short >, [2632](#)
  - std::numeric\_limits< signed char >, [2637](#)
  - std::numeric\_limits< unsigned char >, [2642](#)
  - std::numeric\_limits< unsigned int >, [2647](#)
  - std::numeric\_limits< unsigned long >, [2653](#)
  - std::numeric\_limits< unsigned long long >, [2658](#)
  - std::numeric\_limits< unsigned short >, [2663](#)
  - std::numeric\_limits< wchar\_t >, [2668](#)
  - std::piecewise\_constant\_distribution< \_RealType >, [2743](#)
  - std::piecewise\_linear\_distribution< \_RealType >, [2747](#)
  - std::poisson\_distribution< \_IntType >, [2759](#)
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, [2971](#)
  - std::student\_t\_distribution< \_RealType >, [3031](#)
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, [3040](#)
  - std::uniform\_int\_distribution< \_IntType >, [3143](#)
  - std::uniform\_real\_distribution< \_RealType >, [3146](#)
  - std::weibull\_distribution< \_RealType >, [3346](#)
- min\_element
  - Sorting, [72](#)
- min\_element\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, [1012](#)
- min\_exponent
  - std::\_\_numeric\_limits\_base, [790](#)
  - std::numeric\_limits< \_Tp >, [2576](#)
  - std::numeric\_limits< bool >, [2581](#)
  - std::numeric\_limits< char >, [2587](#)
  - std::numeric\_limits< char16\_t >, [2592](#)
  - std::numeric\_limits< char32\_t >, [2597](#)
  - std::numeric\_limits< double >, [2602](#)
  - std::numeric\_limits< float >, [2608](#)
  - std::numeric\_limits< int >, [2613](#)
  - std::numeric\_limits< long >, [2618](#)
  - std::numeric\_limits< long double >, [2623](#)
  - std::numeric\_limits< long long >, [2629](#)
  - std::numeric\_limits< short >, [2634](#)
  - std::numeric\_limits< signed char >, [2639](#)
  - std::numeric\_limits< unsigned char >, [2644](#)
  - std::numeric\_limits< unsigned int >, [2650](#)
  - std::numeric\_limits< unsigned long >, [2655](#)
  - std::numeric\_limits< unsigned long long >, [2660](#)
  - std::numeric\_limits< unsigned short >, [2665](#)
  - std::numeric\_limits< wchar\_t >, [2671](#)



- min\_exponent10
  - std::\_\_numeric\_limits\_base, 790
  - std::numeric\_limits< \_Tp >, 2576
  - std::numeric\_limits< bool >, 2582
  - std::numeric\_limits< char >, 2587
  - std::numeric\_limits< char16\_t >, 2592
  - std::numeric\_limits< char32\_t >, 2597
  - std::numeric\_limits< double >, 2603
  - std::numeric\_limits< float >, 2608
  - std::numeric\_limits< int >, 2613
  - std::numeric\_limits< long >, 2618
  - std::numeric\_limits< long double >, 2624
  - std::numeric\_limits< long long >, 2629
  - std::numeric\_limits< short >, 2634
  - std::numeric\_limits< signed char >, 2639
  - std::numeric\_limits< unsigned char >, 2645
  - std::numeric\_limits< unsigned int >, 2650
  - std::numeric\_limits< unsigned long >, 2655
  - std::numeric\_limits< unsigned long long >, 2660
  - std::numeric\_limits< unsigned short >, 2666
  - std::numeric\_limits< wchar\_t >, 2671
- minmax
  - Sorting, 72, 73
- minmax\_element
  - Sorting, 73
- minstd\_rand
  - Random Number Generators, 245
- minstd\_rand0
  - Random Number Generators, 245
- minutes
  - Time, 365
- mismatch
  - Non-Mutating, 55, 56
- mod\_based\_range\_hashing.hpp, 3636
- modulus
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2323
- mofunc\_impl.h, 3424
- monetary
  - std::locale, 2356
- money\_get
  - std::money\_get< \_CharT, \_InIter >, 2441
- money\_put
  - std::money\_put< \_CharT, \_OutIter >, 2446
- money\_punct
  - std::money\_punct< \_CharT, \_Intl >, 2451
- monostate.h, 3425
- months
  - Time, 365
- move
  - Mutating, 26
  - Utilities, 302
- move.h, 3425
- move\_backward
  - Mutating, 26
- move\_if\_noexcept
  - Utilities, 303
- move\_only\_function
  - std::move\_only\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 2469, 2470
- mt19937
  - Random Number Generators, 245
- mt19937\_64
  - Random Number Generators, 245
- mt\_allocator.h, 3609
- multiline
  - std::regex\_constants, 718
- multimap
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2478–2480
- multimap.h, 3584
- multiplier
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2323
- multiseq\_partition
  - \_\_gnu\_parallel, 453
- multiseq\_selection
  - \_\_gnu\_parallel, 453
- multiseq\_selection.h, 3733
- multiset
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2506–2508
- multiset.h, 3585
- multiway\_merge
  - \_\_gnu\_parallel, 454
- multiway\_merge.h, 3734
  - \_GLIBCXX\_PARALLEL\_LENGTH, 3736
- multiway\_merge\_3\_variant
  - \_\_gnu\_parallel, 455
- multiway\_merge\_4\_variant
  - \_\_gnu\_parallel, 456
- multiway\_merge\_exact\_splitting
  - \_\_gnu\_parallel, 456
- multiway\_merge\_loser\_tree
  - \_\_gnu\_parallel, 457
- multiway\_merge\_loser\_tree\_sentinel
  - \_\_gnu\_parallel, 457
- multiway\_merge\_loser\_tree\_unguarded
  - \_\_gnu\_parallel, 458
- multiway\_merge\_minimal\_k
  - \_\_gnu\_parallel::Settings, 1012
- multiway\_merge\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1012
- multiway\_merge\_oversampling
  - \_\_gnu\_parallel::Settings, 1012
- multiway\_merge\_sampling\_splitting
  - \_\_gnu\_parallel, 458

- multiway\_merge\_sentinels
  - \_\_gnu\_parallel, 459
- multiway\_mergesort.h, 3736
- Mutating, 19
  - copy, 21
  - copy\_backward, 21
  - copy\_if, 22
  - copy\_n, 22
  - fill, 23
  - fill\_n, 23
  - generate, 24
  - generate\_n, 24
  - is\_partitioned, 25
  - iter\_swap, 25
  - move, 26
  - move\_backward, 26
  - partition, 27
  - partition\_copy, 27
  - partition\_point, 29
  - random\_shuffle, 29, 30
  - remove, 30
  - remove\_copy, 31
  - remove\_copy\_if, 31
  - remove\_if, 32
  - replace, 32
  - replace\_copy\_if, 33
  - replace\_if, 33
  - reverse, 34
  - reverse\_copy, 34
  - rotate, 35
  - rotate\_copy, 35
  - shuffle, 36
  - stable\_partition, 36
  - swap\_ranges, 37
  - transform, 37, 38
  - unique, 38, 40
  - unique\_copy, 40, 41
- mutex, 3707
- Mutexes, 116
  - adopt\_lock, 118
  - call\_once, 117
  - defer\_lock, 118
  - lock, 117
  - try\_lock, 117
  - try\_to\_lock, 118
- name
  - \_\_gnu\_debug::type\_info, 3132
  - std::error\_category, 2113
  - std::locale, 2354
  - std::type\_info, 3133
- nanoseconds
  - Time, 365
- narrow
  - std::\_\_ctype\_abstract\_base< \_CharT >, 757, 758
  - std::basic\_fstream< \_CharT, \_Traits >, 1131
  - std::basic\_ifstream< \_CharT, \_Traits >, 1181
  - std::basic\_ios< \_CharT, \_Traits >, 1214
  - std::basic\_iostream< \_CharT, \_Traits >, 1242
  - std::basic\_istream< \_CharT, \_Traits >, 1289
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2879
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1327
  - std::basic\_ofstream< \_CharT, \_Traits >, 1361
  - std::basic\_ostream< \_CharT, \_Traits >, 1393
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2907
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1427
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1687
  - std::ctype< \_CharT >, 1918, 1919
  - std::ctype< char >, 1944–1946
  - std::ctype< wchar\_t >, 1972, 1973
  - std::ctype\_byname< \_CharT >, 1986
  - std::ctype\_byname< char >, 2005, 2006
- native\_handle
  - std::thread, 3052
- neg\_format
  - std::moneypunct< \_CharT, \_Intl >, 2455
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2463
- negative\_sign
  - std::moneypunct< \_CharT, \_Intl >, 2456
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2463
- Negators, 316
  - not1, 317
  - not2, 317
- nested\_exception
  - std::nested\_exception, 2532
- nested\_exception.h, 3426
- nested\_ptr
  - std::nested\_exception, 2532
- new, 3804
  - operator new, 3805
- new\_allocator.h, 3426, 3427
- new\_delete\_resource
  - Polymorphic memory resources, 345
- new\_handler
  - std, 579
- next\_permutation
  - Sorting, 74
- noboolalpha
  - std, 601
- node.hpp, 3639, 3640
- node\_begin
  - \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, 2696
  - \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >, 2733

- \_\_gnu\_pbds::detail::rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [2803](#)
  - \_\_gnu\_pbds::detail::splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [2982](#)
- node\_const\_iterator
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, Node, \_Alloc >, [1731](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, Node, \_Alloc >, [1732](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [3099](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [3100](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [3102](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [3102](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [3104](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [3105](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [3120](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [3121](#)
- node\_end
  - \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [2696](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >, [2734](#)
  - \_\_gnu\_pbds::detail::rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [2803](#)
  - \_\_gnu\_pbds::detail::splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [2982](#)
- node\_handle.h, [3427](#)
- node\_iterators.hpp, [3611](#)
- node\_metadata\_selector.hpp, [3649](#), [3650](#)
- node\_type
  - \_\_gnu\_pbds::detail::pat\_trie\_base, [2731](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >, [2733](#)
- node\_update
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [3120](#)
- Non-Mutating, [42](#)
  - adjacent\_find, [44](#)
  - all\_of, [44](#)
  - any\_of, [45](#)
  - count, [45](#)
  - count\_if, [46](#)
  - equal, [46](#), [47](#)
  - find, [48](#)
  - find\_end, [48](#), [49](#)
  - find\_first\_of, [50](#)
  - find\_if, [51](#)
  - find\_if\_not, [51](#)
  - for\_each, [52](#)
  - for\_each\_n, [52](#)
  - is\_permutation, [53](#), [54](#)
  - mismatch, [55](#), [56](#)
  - none\_of, [57](#)
  - search, [57](#), [58](#)
  - search\_n, [58](#), [59](#)
- none
  - std::bitset< \_Nb >, [1765](#)
  - std::locale, [2356](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2076](#)
- none\_of
  - Non-Mutating, [57](#)
- norm
  - Complex Numbers, [182](#)
  - std::tr1, [726](#)
- Normal Distributions, [237](#)
  - operator<<, [238](#)
  - operator>>, [238](#)
- normal\_distribution
  - std::normal\_distribution< \_RealType >, [2536](#)
- noshowbase
  - std, [602](#)
- noshowpoint
  - std, [602](#)
- noshowpos
  - std, [602](#)
- noskipws
  - std, [602](#)
- nosubs
  - std::regex\_constants, [718](#)
- not1
  - Negators, [317](#)
- not2
  - Negators, [317](#)
- not\_fn
  - std::experimental, [702](#)
- notify\_cleared

[illegible]

- numbers, [3707](#)
- numeric, [3707](#), [3708](#), [3710](#)
  - std::locale, [2357](#)
- Numeric Arrays, [216](#)
  - ~gslice, [221](#)
  - apply, [221](#), [222](#)
  - begin, [222](#)
  - cshift, [222](#)
  - end, [223](#)
  - gslice, [218](#)
  - gslice\_array, [219](#)
  - indirect\_array, [219](#)
  - mask\_array, [219](#)
  - max, [223](#)
  - min, [223](#)
  - operator=, [223–227](#)
  - operator[], [227–230](#)
  - resize, [231](#)
  - shift, [231](#)
  - size, [232](#)
  - slice, [219](#)
  - slice\_array, [220](#)
  - start, [232](#)
  - stride, [232](#)
  - sum, [232](#)
  - swap, [232](#)
  - valarray, [220](#), [221](#)
- numeric\_traits.h, [3602](#)
- numeric\_fwd.h, [3737](#)
- Numerics, [176](#)
- num\_punct
  - std::num\_punct<\_CharT>, [2673](#), [2674](#)
- oct
  - std, [602](#)
  - std::basic\_fstream<\_CharT, \_Traits>, [1158](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [1201](#)
  - std::basic\_ios<\_CharT, \_Traits>, [1224](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [1270](#)
  - std::basic\_istream<\_CharT, \_Traits>, [1307](#)
  - std::basic\_istream<\_CharT, \_Traits>::sentry, [2895](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1347](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [1381](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [1411](#)
  - std::basic\_ostream<\_CharT, \_Traits>::sentry, [2923](#)
  - std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc>, [1445](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1715](#)
  - std::ios\_base, [2269](#)
- off\_type
  - std::basic\_ios<\_CharT, \_Traits>, [1208](#)
  - std::basic\_streambuf<\_CharT, \_Traits>, [1459](#)
  - std::wbuffer\_convert<\_Codecvt, \_Elem, \_Tr>, [3330](#)
- ofstream
  - I/O, [165](#)
- omp\_loop.h, [3738](#)
- omp\_loop\_static.h, [3739](#)
- once\_flag
  - std::once\_flag, [2684](#)
- open
  - \_\_gnu\_cxx::enc\_filebuf<\_CharT>, [2089](#), [2090](#)
  - \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits>, [2994–2996](#)
  - std::basic\_filebuf<\_CharT, \_Traits>, [1096](#), [1097](#)
  - std::basic\_fstream<\_CharT, \_Traits>, [1131](#), [1132](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [1182](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [1363](#)
- openmode
  - std::basic\_fstream<\_CharT, \_Traits>, [1118](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [1169](#)
  - std::basic\_ios<\_CharT, \_Traits>, [1208](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [1231](#)
  - std::basic\_istream<\_CharT, \_Traits>, [1277](#)
  - std::basic\_istream<\_CharT, \_Traits>::sentry, [2868](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1316](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [1354](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [1387](#)
  - std::basic\_ostream<\_CharT, \_Traits>::sentry, [2902](#)
  - std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc>, [1419](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1676](#)
  - std::ios\_base, [2260](#)
- operator\_iterator
  - \_\_gnu\_debug::Safe\_iterator<\_Iterator, \_Sequence, \_Category>, [967](#)
  - \_\_gnu\_debug::Safe\_local\_iterator<\_Iterator, \_UContainer>, [981](#)
- operator\_RAlter
  - \_\_gnu\_parallel::GuardedIterator<\_RAIter, \_Compare>, [887](#)
- operator\_bool
  - std::basic\_fstream<\_CharT, \_Traits>, [1132](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [1183](#)
  - std::basic\_ios<\_CharT, \_Traits>, [1215](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [1243](#)
  - std::basic\_istream<\_CharT, \_Traits>, [1289](#)
  - std::basic\_istream<\_CharT, \_Traits>::sentry, [2880](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1327](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [1364](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [1394](#)
  - std::basic\_ostream<\_CharT, \_Traits>::sentry, [2908](#)
  - std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc>, [1427](#)

- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1688
- std::copyable\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 1907
- std::error\_code, 2114
- std::error\_condition, 2116
- std::function< \_Res(\_ArgTypes...)>, 2167
- std::move\_only\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 2470
- std::shared\_ptr< \_Tp >, 2966
- std::tr2::bool\_set, 1771
- std::unique\_ptr< \_Tp, \_Dp >, 3152
- std::unique\_ptr< \_Tp[], \_Dp >, 3159
- operator new
  - new, 3805
- operator streamoff
  - std::fpos< \_StateT >, 2159
- operator string\_type
  - std::sub\_match< \_Bilter >, 3036
- operator!
  - std::basic\_fstream< \_CharT, \_Traits >, 1132
  - std::basic\_ifstream< \_CharT, \_Traits >, 1183
  - std::basic\_ios< \_CharT, \_Traits >, 1215
  - std::basic\_iostream< \_CharT, \_Traits >, 1243
  - std::basic\_istream< \_CharT, \_Traits >, 1289
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2880
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1327
  - std::basic\_ofstream< \_CharT, \_Traits >, 1364
  - std::basic\_ostream< \_CharT, \_Traits >, 1394
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2908
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1427
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1688
  - std::valarray< \_Tp >, 3282
- operator!=
  - \_\_gnu\_cxx, 395
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1725
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1729
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1739
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1743
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, 2306, 2307
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, 2311
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 936
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 939
  - \_\_gnu\_pbds::dynamic\_bitset, 135
  - std, 602, 603
  - std::rel\_ops, 719
- operator<
  - \_\_gnu\_cxx, 398
  - \_\_gnu\_parallel::GuardedIterator< \_RAIter, \_Compare >, 887
  - Pointer Abstractions, 338, 339
  - std, 605–607
  - Time, 373
- operator<<
  - Bernoulli Distributions, 235
  - Complex Numbers, 185
  - Diagnostics, 125
  - Dynamic Bitset, 136
  - Normal Distributions, 238
  - Pointer Abstractions, 339
  - Poisson Distributions, 239, 240
  - Random Number Generators, 246
  - Regular Expressions, 256
  - std, 608–617
  - std::\_\_detail, 656
  - std::basic\_fstream< \_CharT, \_Traits >, 1132–1134, 1136–1138
  - std::basic\_iostream< \_CharT, \_Traits >, 1243, 1244, 1246–1249
  - std::basic\_ofstream< \_CharT, \_Traits >, 1364–1369
  - std::basic\_ostream< \_CharT, \_Traits >, 1394, 1395, 1397–1400
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2908–2913
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1428–1432
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1688, 1689, 1691–1694
  - std::binomial\_distribution< \_IntType >, 1750
  - std::bitset< \_Nb >, 1766
  - std::chi\_squared\_distribution< \_RealType >, 1799
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2060
  - std::discrete\_distribution< \_IntType >, 2063
  - std::filesystem::path, 2741
  - std::fisher\_f\_distribution< \_RealType >, 2134
  - std::gamma\_distribution< \_RealType >, 2182
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2322
  - std::lognormal\_distribution< \_RealType >, 2367
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 2427
  - std::negative\_binomial\_distribution< \_IntType >, 2531



- std::normal\_distribution< \_RealType >, [2538](#)
- std::piecewise\_constant\_distribution< \_RealType >, [2745](#)
- std::piecewise\_linear\_distribution< \_RealType >, [2748](#)
- std::poisson\_distribution< \_IntType >, [2760](#)
- std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, [2971](#)
- std::student\_t\_distribution< \_RealType >, [3032](#)
- std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, [3041](#)
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2077](#)
- std::unique\_ptr< \_Tp[], \_Dp >, [3162](#)
- Time, [373](#)
- Uniform Distributions, [243](#)
- operator<=<=
  - std::bitset< \_Nb >, [1766](#)
  - std::gslice\_array< \_Tp >, [2206](#)
  - std::indirect\_array< \_Tp >, [2247](#)
  - std::mask\_array< \_Tp >, [2406](#)
  - std::slice\_array< \_Tp >, [2975](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2077](#)
  - std::valarray< \_Tp >, [3284](#)
- operator<=
  - \_\_gnu\_cxx, [399](#), [400](#)
  - \_\_gnu\_parallel::GuardedIterator< \_RAIter, \_Compare >, [887](#)
  - Dynamic Bitset., [136](#)
  - Pointer Abstractions, [339](#), [340](#)
  - std, [617](#), [618](#)
  - std::rel\_ops, [719](#)
  - Time, [374](#)
- operator<=>
  - \_\_gnu\_parallel::IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, [896](#), [897](#)
  - Diagnostics, [125](#)
  - Regular Expressions, [256](#), [257](#)
  - std, [618](#)–[620](#)
  - std::error\_category, [2113](#)
  - std::filesystem::path, [2741](#)
  - Time, [374](#)
  - Utilities, [304](#)
- operator>
  - \_\_gnu\_cxx, [401](#), [402](#)
  - Dynamic Bitset., [136](#)
  - Pointer Abstractions, [340](#), [341](#)
  - std, [626](#), [627](#)
  - std::rel\_ops, [719](#)
  - Time, [374](#)
- operator>>
  - Bernoulli Distributions, [235](#), [237](#)
  - Complex Numbers, [185](#)
  - Dynamic Bitset., [137](#)
  - Normal Distributions, [238](#)
  - Poisson Distributions, [240](#), [241](#)
  - std, [627](#)–[632](#)
  - std::\_\_detail, [656](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1138](#)–[1143](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1183](#)–[1188](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1249](#), [1250](#), [1252](#)–[1255](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1289](#)–[1294](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2880](#)–[2885](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1328](#), [1329](#), [1331](#)–[1333](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1694](#), [1695](#), [1697](#)–[1700](#)
  - std::binomial\_distribution< \_IntType >, [1752](#)
  - std::bitset< \_Nb >, [1766](#)
  - std::chi\_squared\_distribution< \_RealType >, [1799](#)
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2060](#)
  - std::discrete\_distribution< \_IntType >, [2064](#)
  - std::filesystem::path, [2741](#)
  - std::fisher\_f\_distribution< \_RealType >, [2135](#)
  - std::gamma\_distribution< \_RealType >, [2183](#)
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, [2245](#)
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [2322](#)
  - std::lognormal\_distribution< \_RealType >, [2368](#)
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [2427](#)
  - std::negative\_binomial\_distribution< \_IntType >, [2531](#)
  - std::normal\_distribution< \_RealType >, [2538](#)
  - std::piecewise\_constant\_distribution< \_RealType >, [2745](#)
  - std::piecewise\_linear\_distribution< \_RealType >, [2749](#)
  - std::poisson\_distribution< \_IntType >, [2761](#)
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, [2972](#)
  - std::student\_t\_distribution< \_RealType >, [3033](#)
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, [3041](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2078](#)
  - Uniform Distributions, [243](#), [244](#)
- operator>>=
  - std::bitset< \_Nb >, [1766](#)
  - std::gslice\_array< \_Tp >, [2206](#)
  - std::indirect\_array< \_Tp >, [2247](#)
  - std::mask\_array< \_Tp >, [2406](#)
  - std::slice\_array< \_Tp >, [2975](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2078](#)
  - std::valarray< \_Tp >, [3284](#)

- operator>=
  - \_\_gnu\_cxx, 402, 403
  - Dynamic Bitset., 136
  - Pointer Abstractions, 341
  - std, 627
  - std::rel\_ops, 720
  - Time, 374
- operator()
  - \_\_gnu\_cxx::subtractive\_rng, 3043
  - \_\_gnu\_parallel::Nothing, 941
  - \_\_gnu\_parallel::RandomNumber, 952
  - \_\_gnu\_parallel::\_\_accumulate\_selector<\_It>, 730
  - \_\_gnu\_parallel::\_\_adjacent\_find\_selector, 732
  - \_\_gnu\_parallel::\_\_count\_if\_selector<\_It, \_Diff>, 747
  - \_\_gnu\_parallel::\_\_count\_selector<\_It, \_Diff>, 748
  - \_\_gnu\_parallel::\_\_fill\_selector<\_It>, 765
  - \_\_gnu\_parallel::\_\_find\_first\_of\_selector<\_FIterator>, 766
  - \_\_gnu\_parallel::\_\_find\_if\_selector, 768
  - \_\_gnu\_parallel::\_\_for\_each\_selector<\_It>, 769
  - \_\_gnu\_parallel::\_\_generate\_selector<\_It>, 770
  - \_\_gnu\_parallel::\_\_identity\_selector<\_It>, 774
  - \_\_gnu\_parallel::\_\_inner\_product\_selector<\_It, \_It2, \_Tp>, 775
  - \_\_gnu\_parallel::\_\_mismatch\_selector, 779
  - \_\_gnu\_parallel::\_\_replace\_if\_selector<\_It, \_Op, \_Tp>, 801
  - \_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp>, 803
  - \_\_gnu\_parallel::\_\_transform1\_selector<\_It>, 804
  - \_\_gnu\_parallel::\_\_transform2\_selector<\_It>, 805
  - \_\_gnu\_pbds::direct\_mask\_range\_hashing<Size\_Type>, 2052
  - \_\_gnu\_pbds::direct\_mod\_range\_hashing<Size\_Type>, 2053
  - \_\_gnu\_pbds::linear\_probe\_fn<Size\_Type>, 2324
  - \_\_gnu\_pbds::lu\_counter\_policy<Max\_Count, \_Alloc>, 2370
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy<\_Alloc>, 2374
  - \_\_gnu\_pbds::quadratic\_probe\_fn<Size\_Type>, 2774
  - \_\_gnu\_pbds::sample\_probe\_fn, 2844
  - \_\_gnu\_pbds::sample\_range\_hashing, 2845
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 2846
  - \_\_gnu\_pbds::sample\_trie\_node\_update<Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc>, 2855
  - \_\_gnu\_pbds::sample\_update\_policy, 2856
  - \_\_gnu\_pbds::tree\_order\_statistics\_node\_update<Node\_Cltr, Node\_Itr, Cmp\_Fn, \_Alloc>, 3097
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update<Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc>, 3111
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update<Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc>, 3115
  - std::bernoulli\_distribution, 1718
  - std::binomial\_distribution<\_IntType>, 1749
  - std::cauchy\_distribution<\_RealType>, 1775
  - std::chi\_squared\_distribution<\_RealType>, 1798
  - std::copyable\_function<\_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 1907
  - std::default\_delete<\_Tp>, 2020
  - std::default\_delete<\_Tp[]>, 2021
  - std::discard\_block\_engine<\_RandomNumberEngine, \_\_p, \_\_r>, 2059
  - std::discrete\_distribution<\_IntType>, 2062
  - std::exponential\_distribution<\_RealType>, 2122
  - std::extreme\_value\_distribution<\_RealType>, 2125
  - std::fisher\_f\_distribution<\_RealType>, 2134
  - std::function<\_Res(\_ArgTypes...)>, 2167
  - std::function\_ref<\_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 2173
  - std::gamma\_distribution<\_RealType>, 2181, 2182
  - std::geometric\_distribution<\_IntType>, 2185
  - std::independent\_bits\_engine<\_RandomNumberEngine, \_\_w, \_UIntType>, 2244
  - std::linear\_congruential\_engine<\_UIntType, \_\_a, \_\_c, \_\_m>, 2321
  - std::locale, 2354
  - std::lognormal\_distribution<\_RealType>, 2366
  - std::move\_only\_function<\_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 2470
  - std::negative\_binomial\_distribution<\_IntType>, 2530
  - std::normal\_distribution<\_RealType>, 2537
  - std::piecewise\_constant\_distribution<\_RealType>, 2743
  - std::piecewise\_linear\_distribution<\_RealType>, 2748
  - std::plus<\_Tp>, 2751
  - std::poisson\_distribution<\_IntType>, 2759, 2760
  - std::shuffle\_order\_engine<\_RandomNumberEngine, \_\_k>, 2971
  - std::student\_t\_distribution<\_RealType>, 3031
  - std::subtract\_with\_carry\_engine<\_UIntType, \_\_w, \_\_s, \_\_r>, 3040
  - std::uniform\_int\_distribution<\_IntType>, 3143
  - std::uniform\_real\_distribution<\_RealType>, 3146
  - std::weibull\_distribution<\_RealType>, 3346
- operator+
  - \_\_gnu\_cxx, 396, 397
  - Complex Numbers, 183
  - std, 604, 605
  - std::fpos<\_StateT>, 2159
  - std::reverse\_iterator<\_Iterator>, 2835
  - std::valarray<\_Tp>, 3283
  - Time, 372
- operator++



- `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 967, 968
- `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 982
- `__gnu_parallel::GuardedIterator< _RAIter, _Compare >`, 887
- `std::back_insert_iterator< _Container >`, 1078
- `std::front_insert_iterator< _Container >`, 2164
- `std::insert_iterator< _Container >`, 2253
- `std::istreambuf_iterator< _CharT, _Traits >`, 2296
- `std::ostreambuf_iterator< _CharT, _Traits >`, 2692
- `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 2817, 2818
- `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 2822
- `std::reverse_iterator< _Iterator >`, 2835
- `operator+=`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 841, 843
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1526
  - Complex Numbers, 183
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1615–1617
  - `std::complex< _Tp >`, 1867
  - `std::complex< double >`, 1869
  - `std::complex< float >`, 1872
  - `std::complex< long double >`, 1875
  - `std::fpos< _StateT >`, 2159
  - `std::gslice_array< _Tp >`, 2205
  - `std::indirect_array< _Tp >`, 2247
  - `std::mask_array< _Tp >`, 2406
  - `std::reverse_iterator< _Iterator >`, 2836
  - `std::slice_array< _Tp >`, 2975
  - `std::valarray< _Tp >`, 3283
- `operator-`
  - Complex Numbers, 183, 184
  - Dynamic Bitset., 135
  - `std::fpos< _StateT >`, 2159, 2160
  - `std::reverse_iterator< _Iterator >`, 2836
  - `std::valarray< _Tp >`, 3283
  - Time, 372
- `operator->`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 968
  - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 982
  - `__gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 1740
  - `__gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 1743
  - `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >`, 2307
  - `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >`, 2311
- `std::auto_ptr< _Tp >`, 1074
- `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 2818
- `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 2822
- `std::reverse_iterator< _Iterator >`, 2837
- `std::unique_ptr< _Tp, _Dp >`, 3152
- `std::unique_ptr< _Tp[], _Dp >`, 3160
- `operator--`
  - `std::reverse_iterator< _Iterator >`, 2836
- `operator=`
  - Complex Numbers, 184
  - `std::complex< _Tp >`, 1867
  - `std::complex< double >`, 1869
  - `std::complex< float >`, 1872
  - `std::complex< long double >`, 1875
  - `std::fpos< _StateT >`, 2160
  - `std::gslice_array< _Tp >`, 2205
  - `std::indirect_array< _Tp >`, 2247
  - `std::mask_array< _Tp >`, 2406
  - `std::reverse_iterator< _Iterator >`, 2836
  - `std::slice_array< _Tp >`, 2975
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2077
  - `std::valarray< _Tp >`, 3283
- `operator/`
  - Complex Numbers, 184
  - `std::filesystem::path`, 2741
  - Time, 373
- `operator/=`
  - Complex Numbers, 184
  - `std::complex< double >`, 1869, 1870
  - `std::complex< float >`, 1872
  - `std::complex< long double >`, 1875
  - `std::gslice_array< _Tp >`, 2205
  - `std::indirect_array< _Tp >`, 2247
  - `std::mask_array< _Tp >`, 2406
  - `std::slice_array< _Tp >`, 2975
  - `std::valarray< _Tp >`, 3284
- `operator=`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 844, 845
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 968
  - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 982
  - Complex Numbers, 185
  - Numeric Arrays, 223–227
  - `std::auto_ptr< _Tp >`, 1074, 1075
  - `std::back_insert_iterator< _Container >`, 1078
  - `std::basic_regex< _Ch_type, _Rx_traits >`, 1454
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1617–1620
  - `std::complex< double >`, 1870

- std::complex< float >, [1872](#), [1873](#)
- std::complex< long double >, [1875](#)
- std::copyable\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, [1907](#), [1908](#)
- std::deque< \_Tp, \_Alloc >, [2046](#), [2047](#)
- std::experimental::fundamentals\_v1::any, [1065](#), [1066](#)
- std::forward\_list< \_Tp, \_Alloc >, [2153](#), [2154](#)
- std::front\_insert\_iterator< \_Container >, [2164](#)
- std::function< \_Res(\_ArgTypes...)>, [2168](#), [2169](#)
- std::insert\_iterator< \_Container >, [2254](#)
- std::list< \_Tp, \_Alloc >, [2340](#)
- std::locale, [2354](#)
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2400](#), [2401](#)
- std::match\_results< \_Bi\_iter, \_Alloc >, [2414](#)
- std::move\_only\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, [2470](#), [2471](#)
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [2495](#)
- std::multiset< \_Key, \_Compare, \_Alloc >, [2520](#)
- std::once\_flag, [2684](#)
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [2688](#)
- std::ostreambuf\_iterator< \_CharT, \_Traits >, [2692](#)
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [2818](#)
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [2822](#)
- std::set< \_Key, \_Compare, \_Alloc >, [2948](#)
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2078](#)
- std::unique\_ptr< \_Tp, \_Dp >, [3152](#)
- std::unique\_ptr< \_Tp[], \_Dp >, [3160](#), [3161](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3190](#), [3191](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3221](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3247](#), [3248](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3275](#), [3276](#)
- std::vector< \_Tp, \_Alloc >, [3302](#), [3303](#)
- std::vector< bool, \_Alloc >, [3323](#)
- operator==
  - \_\_gnu\_cxx, [400](#), [401](#)
  - \_\_gnu\_parallel::iterator\_pair< \_Iterator1, \_Iterator2, \_IteratorCategory >, [897](#), [898](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator< Node, Const\_Iterator, Iterator, \_Alloc >, [1725](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator< Node, Const\_Iterator, Iterator, \_Alloc >, [1730](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1740](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1743](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, [2307](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, [2311](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_C\_iterator, Iterator, \_Alloc >, [936](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_C\_iterator, Iterator, \_Alloc >, [939](#)
  - Allocators, [324](#)
  - Complex Numbers, [185](#)
  - Diagnostics, [125](#), [126](#)
  - Iterators, [172](#)
  - Pointer Abstractions, [340](#)
  - Regular Expressions, [257](#), [258](#)
  - std, [621](#)–[626](#)
  - std::\_fwdlist::iterator< \_Const, \_Ptr >, [893](#)
  - std::\_Fwd\_list\_const\_iterator< \_Tp >, [883](#)
  - std::\_Fwd\_list\_iterator< \_Tp >, [884](#)
  - std::bernoulli\_distribution, [1719](#)
  - std::binomial\_distribution< \_IntType >, [1752](#)
  - std::bitset< \_Nb >, [1766](#)
  - std::cauchy\_distribution< \_RealType >, [1776](#)
  - std::chi\_squared\_distribution< \_RealType >, [1799](#)
  - std::copyable\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, [1908](#)
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2060](#)
  - std::discrete\_distribution< \_IntType >, [2063](#)
  - std::error\_category, [2113](#)
  - std::exponential\_distribution< \_RealType >, [2123](#)
  - std::extreme\_value\_distribution< \_RealType >, [2126](#)
  - std::filesystem::path, [2741](#)
  - std::fisher\_f\_distribution< \_RealType >, [2135](#)
  - std::gamma\_distribution< \_RealType >, [2183](#)
  - std::geometric\_distribution< \_IntType >, [2185](#)
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, [2244](#)
  - std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, [2292](#)
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [2322](#)
  - std::locale, [2354](#)
  - std::lognormal\_distribution< \_RealType >, [2367](#)
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [2427](#)
  - std::move\_only\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, [2471](#)
  - std::negative\_binomial\_distribution< \_IntType >, [2531](#)
  - std::normal\_distribution< \_RealType >, [2538](#)
  - std::piecewise\_constant\_distribution< \_RealType >, [2541](#)

- 2745
  - std::piecewise\_linear\_distribution< \_RealType >, 2749
  - std::poisson\_distribution< \_IntType >, 2761
  - std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 2818
  - std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 2822
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 2972
  - std::slice, 2973
  - std::student\_t\_distribution< \_RealType >, 3032
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3041
  - std::uniform\_int\_distribution< \_IntType >, 3144
  - std::uniform\_real\_distribution< \_RealType >, 3147
  - std::weibull\_distribution< \_RealType >, 3347
  - Time, 374
  - Utilities, 304
- operator%
  - Time, 371
- operator%=
  - std::gslice\_array< \_Tp >, 2205
  - std::indirect\_array< \_Tp >, 2247
  - std::mask\_array< \_Tp >, 2405
  - std::slice\_array< \_Tp >, 2975
  - std::valarray< \_Tp >, 3282
- operator&
  - Dynamic Bitset., 135
  - std, 603
  - std::regex\_constants, 712
- operator&=
  - std::bitset< \_Nb >, 1766
  - std::gslice\_array< \_Tp >, 2205
  - std::indirect\_array< \_Tp >, 2247
  - std::mask\_array< \_Tp >, 2405
  - std::regex\_constants, 712, 713
  - std::slice\_array< \_Tp >, 2975
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2077
  - std::valarray< \_Tp >, 3282
- operator[]
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 845
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1526, 1527
  - Numeric Arrays, 227–230
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1620, 1621
  - std::bitset< \_Nb >, 1767
  - std::deque< \_Tp, \_Alloc >, 2047, 2048
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2401
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2414
  - std::reverse\_iterator< \_Iterator >, 2837
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2078, 2079
  - std::unique\_ptr< \_Tp[], \_Dp >, 3161
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3191
  - std::vector< \_Tp, \_Alloc >, 3303
  - std::vector< bool, \_Alloc >, 3323, 3324
- operator\*
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 967
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_UContainer >, 981
  - \_\_gnu\_parallel::GuardedIterator< \_RAIter, \_Compare >, 887
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, 1725
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, 1730
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 1740
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 1743
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_< Node, \_Alloc >, 2307
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, \_Alloc >, 2311
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, 2700
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer\_< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 936
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter\_< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 939
  - Complex Numbers, 182
  - std::auto\_ptr< \_Tp >, 1074
  - std::back\_insert\_iterator< \_Container >, 1078
  - std::front\_insert\_iterator< \_Container >, 2164
  - std::insert\_iterator< \_Container >, 2253
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 2296
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 2692
  - std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 2817
  - std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 2822
  - std::reverse\_iterator< \_Iterator >, 2835
  - std::unique\_ptr< \_Tp, \_Dp >, 3152
  - std::unique\_ptr< \_Tp[], \_Dp >, 3160
  - Time, 371
- operator\*=
  - Complex Numbers, 182
  - std::complex< double >, 1869
  - std::complex< float >, 1872
  - std::complex< long double >, 1874

- std::gslice\_array< \_Tp >, 2205
- std::indirect\_array< \_Tp >, 2247
- std::mask\_array< \_Tp >, 2406
- std::slice\_array< \_Tp >, 2975
- std::valarray< \_Tp >, 3283
- operator~
  - std::bitset< \_Nb >, 1768
  - std::regex\_constants, 714
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2080
  - std::valarray< \_Tp >, 3285
- operator^
  - Dynamic Bitset., 137
  - std, 632
  - std::regex\_constants, 713
- operator^=
  - std::bitset< \_Nb >, 1767
  - std::gslice\_array< \_Tp >, 2206
  - std::indirect\_array< \_Tp >, 2248
  - std::mask\_array< \_Tp >, 2406
  - std::regex\_constants, 713
  - std::slice\_array< \_Tp >, 2976
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2079
  - std::valarray< \_Tp >, 3284
- operator|
  - Dynamic Bitset., 137
  - std, 633
  - std::regex\_constants, 714
- operator|=
  - std::bitset< \_Nb >, 1768
  - std::gslice\_array< \_Tp >, 2206
  - std::indirect\_array< \_Tp >, 2248
  - std::mask\_array< \_Tp >, 2406
  - std::regex\_constants, 714
  - std::slice\_array< \_Tp >, 2976
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2079
  - std::valarray< \_Tp >, 3285
- opt\_random.h, 3799
- optimize
  - std::regex\_constants, 718
- optional, 3712
- Optional values, 286
  - in\_place, 287
  - nullopt, 287
- order\_of\_key
  - \_\_gnu\_pbds::tree\_order\_statistics\_node\_update< Node\_Cltr, Node\_Itr, Cmp\_Fn, \_Alloc >, 3097
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, 3111
- order\_of\_prefix
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, 3111
- order\_preserving
  - \_\_gnu\_pbds::container\_traits< Cntnr >, 1899
- order\_statistics\_imp.hpp, 3650
- os\_defines.h, 3799
- ostream, 3712
  - I/O, 165
- ostream.h, 3427
- ostream.tcc, 3429
- ostream\_insert.h, 3429
- ostream\_iterator
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 2688
- ostream\_type
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 2687
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 2690
- ostreambuf\_iterator
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 2691
- ostringstream
  - I/O, 165
- out
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, 744
  - std::basic\_fstream< \_CharT, \_Traits >, 1158
  - std::basic\_ifstream< \_CharT, \_Traits >, 1201
  - std::basic\_ios< \_CharT, \_Traits >, 1224
  - std::basic\_iostream< \_CharT, \_Traits >, 1270
  - std::basic\_istream< \_CharT, \_Traits >, 1307
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2895
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1347
  - std::basic\_ofstream< \_CharT, \_Traits >, 1381
  - std::basic\_ostream< \_CharT, \_Traits >, 1411
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2923
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1445
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1715
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 1803
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1811, 1812
  - std::codecvt< char, char, mbstate\_t >, 1820
  - std::codecvt< char16\_t, char, mbstate\_t >, 1828, 1829
  - std::codecvt< char32\_t, char, mbstate\_t >, 1837, 1838
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1847
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1854
  - std::ios\_base, 2269
- out\_ptr.h, 3429
- ov\_tree\_map.hpp, 3641
- overflow
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2090
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2996
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3014
  - std::basic\_filebuf< \_CharT, \_Traits >, 1098
  - std::basic\_streambuf< \_CharT, \_Traits >, 1462

- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1657
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3333
- owner\_before
  - std::shared\_ptr< \_Tp >, 2966, 2967
- p
  - std::bernoulli\_distribution, 1718
  - std::binomial\_distribution< \_IntType >, 1750
  - std::geometric\_distribution< \_IntType >, 2185
  - std::negative\_binomial\_distribution< \_IntType >, 2530
- pair
  - \_\_gnu\_cxx::pair< \_T1, \_T2 >, 2709, 2710
  - std::pair< \_T1, \_T2 >, 2713
  - Utilities, 298
- pairing\_heap.hpp, 3642
- par\_loop.h, 3739
- parallel.h, 3739
- parallel\_balanced
  - \_\_gnu\_parallel, 423
- parallel\_multiway\_merge
  - \_\_gnu\_parallel, 460
- parallel\_omp\_loop
  - \_\_gnu\_parallel, 423
- parallel\_omp\_loop\_static
  - \_\_gnu\_parallel, 423
- parallel\_sort\_mwms
  - \_\_gnu\_parallel, 461
- parallel\_sort\_mwms\_pu
  - \_\_gnu\_parallel, 461
- parallel\_tag
  - \_\_gnu\_parallel::parallel\_tag, 2719
- parallel\_taskqueue
  - \_\_gnu\_parallel, 423
- parallel\_unbalanced
  - \_\_gnu\_parallel, 423
- Parallelism TS, 294
- param
  - std::bernoulli\_distribution, 1718
  - std::binomial\_distribution< \_IntType >, 1750
  - std::cauchy\_distribution< \_RealType >, 1775
  - std::chi\_squared\_distribution< \_RealType >, 1798
  - std::discrete\_distribution< \_IntType >, 2062, 2063
  - std::exponential\_distribution< \_RealType >, 2122
  - std::extreme\_value\_distribution< \_RealType >, 2125, 2126
  - std::fisher\_f\_distribution< \_RealType >, 2134
  - std::gamma\_distribution< \_RealType >, 2182
  - std::geometric\_distribution< \_IntType >, 2185
  - std::lognormal\_distribution< \_RealType >, 2367
  - std::negative\_binomial\_distribution< \_IntType >, 2530
  - std::normal\_distribution< \_RealType >, 2537
- std::piecewise\_constant\_distribution< \_RealType >, 2743
- std::piecewise\_linear\_distribution< \_RealType >, 2748
- std::poisson\_distribution< \_IntType >, 2760
- std::student\_t\_distribution< \_RealType >, 3032
- std::uniform\_int\_distribution< \_IntType >, 3144
- std::uniform\_real\_distribution< \_RealType >, 3146
- std::weibull\_distribution< \_RealType >, 3347
- parse\_numbers.h, 3429
- partial\_sort
  - Sorting, 76
- partial\_sort\_copy
  - Sorting, 77
- partial\_sort\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1013
- partial\_sum
  - Generalized Numeric operations, 12, 13
- partial\_sum.h, 3740
- partial\_sum\_dilation
  - \_\_gnu\_parallel::Settings, 1013
- partial\_sum\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1013
- partition
  - Mutating, 27
- partition.h, 3740
  - \_GLIBCXX\_VOLATILE, 3740
- partition\_chunk\_share
  - \_\_gnu\_parallel::Settings, 1013
- partition\_chunk\_size
  - \_\_gnu\_parallel::Settings, 1013
- partition\_copy
  - Mutating, 27
- partition\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1013
- partition\_point
  - Mutating, 29
- pat\_trie.hpp, 3642
- pat\_trie\_base.hpp, 3643
- pbackfail
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2091
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2997
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3014
  - std::basic\_filebuf< \_CharT, \_Traits >, 1098
  - std::basic\_streambuf< \_CharT, \_Traits >, 1463
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1657
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3334
- pbase
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2091
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2997
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3015
  - std::basic\_filebuf< \_CharT, \_Traits >, 1099

- std::basic\_streambuf< \_CharT, \_Traits >, 1463
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1658
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3334
- pbump
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2092
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2997
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3015
- std::basic\_filebuf< \_CharT, \_Traits >, 1099
- std::basic\_streambuf< \_CharT, \_Traits >, 1464
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1658
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3334
- peek
  - std::basic\_fstream< \_CharT, \_Traits >, 1144
  - std::basic\_ifstream< \_CharT, \_Traits >, 1188
  - std::basic\_iostream< \_CharT, \_Traits >, 1256
  - std::basic\_istream< \_CharT, \_Traits >, 1295
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2885
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1334
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1701
- perm\_options
  - Filesystem, 162
- perms
  - Filesystem, 162
  - Filesystem TS, 279
- piecewise\_construct
  - Utilities, 306
- pod\_char\_traits.h, 3657
- point\_const\_iterator.hpp, 3624
- point\_iterator.hpp, 3653
- point\_iterators.hpp, 3611
- pointer
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1739
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1742
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, 2306
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_pointer\_const\_iterator< Node, \_Alloc >, 2310
  - std::allocator\_traits< \_Alloc >, 1033
  - std::allocator\_traits< allocator< \_Tp > >, 1039
  - std::allocator\_traits< allocator< void > >, 1048
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, 1056
  - std::back\_insert\_iterator< \_Container >, 1077
  - std::front\_insert\_iterator< \_Container >, 2163
  - std::insert\_iterator< \_Container >, 2253
  - std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2292
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 2295
  - std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 2300
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 2687
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 2691
  - std::pointer\_traits< \_Ptr >, 2754
  - std::pointer\_traits< \_Tp \* >, 2756
  - std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 2796
  - std::set< \_Key, \_Compare, \_Alloc >, 2932
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3171
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3202
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3230
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3257
- Pointer Abstractions, 325
  - allocate\_shared, 328
  - atomic\_compare\_exchange\_strong, 329
  - atomic\_compare\_exchange\_strong\_explicit, 330
  - atomic\_compare\_exchange\_weak, 331
  - atomic\_compare\_exchange\_weak\_explicit, 332
  - atomic\_exchange, 333
  - atomic\_exchange\_explicit, 333, 334
  - atomic\_is\_lock\_free, 334
  - atomic\_load, 335
  - atomic\_load\_explicit, 335, 336
  - atomic\_store, 336
  - atomic\_store\_explicit, 337
  - const\_pointer\_cast, 337
  - dynamic\_pointer\_cast, 338
  - get\_deleter, 338
  - make\_shared, 338
  - operator<, 338, 339
  - operator<=, 339
  - operator<=, 339, 340
  - operator>, 340, 341
  - operator>=, 341
  - operator==, 340
  - reinterpret\_pointer\_cast, 341
  - static\_pointer\_cast, 342
  - swap, 342
  - to\_address, 342, 343
- Pointer Safety and Garbage Collection, 343
  - declare\_no\_pointers, 344
  - declare\_reachable, 344
  - get\_pointer\_safety, 344
  - pointer\_safety, 344
  - undeclare\_no\_pointers, 344
  - undeclare\_reachable, 344
- pointer.h, 3657
- pointer\_safety
  - Pointer Safety and Garbage Collection, 344



- pointer\_to
  - std::pointer\_traits< \_Ptr >, 2755
  - std::pointer\_traits< \_Tp \* >, 2757
- Poisson Distributions, 239
  - operator<<, 239, 240
  - operator>>, 240, 241
- polar
  - Complex Numbers, 185
  - std::tr1, 726
- Policy-Based Data Structures, 138
- policy\_access\_fnimps.hpp, 3625
- Polymorphic memory resources, 345
  - new\_delete\_resource, 345
- pool\_allocator.h, 3659
- pop
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, 2767
  - std::queue< \_Tp, \_Sequence >, 2777
  - std::stack< \_Tp, \_Sequence >, 2986
- pop\_back
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 846
  - \_\_gnu\_parallel::RestrictedBoundedConcurrentQueue< \_Tp >, 955
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1621
  - std::deque< \_Tp, \_Alloc >, 2048
  - std::list< \_Tp, \_Alloc >, 2341
  - std::vector< \_Tp, \_Alloc >, 3304
  - std::vector< bool, \_Alloc >, 3324
- pop\_front
  - \_\_gnu\_parallel::RestrictedBoundedConcurrentQueue< \_Tp >, 955
  - std::deque< \_Tp, \_Alloc >, 2048
  - std::forward\_list< \_Tp, \_Alloc >, 2154
  - std::list< \_Tp, \_Alloc >, 2341
- pop\_heap
  - Heap, 89, 90
- pos\_format
  - std::moneypunct< \_CharT, \_Intl >, 2456
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2464
- pos\_type
  - std::basic\_ios< \_CharT, \_Traits >, 1208
  - std::basic\_streambuf< \_CharT, \_Traits >, 1459
  - std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, 3330
- position
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2414
- positive\_sign
  - std::moneypunct< \_CharT, \_Intl >, 2456
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2464
- postypes.h, 3430
- pow
  - Complex Numbers, 186
  - math.h, 3701
- power
  - SGL, 155
- pptr
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2092
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2998
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3015
  - std::basic\_filebuf< \_CharT, \_Traits >, 1099
  - std::basic\_streambuf< \_CharT, \_Traits >, 1464
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1658
  - std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, 3335
- precision
  - std::basic\_fstream< \_CharT, \_Traits >, 1144
  - std::basic\_ifstream< \_CharT, \_Traits >, 1188, 1189
  - std::basic\_ios< \_CharT, \_Traits >, 1215
  - std::basic\_iostream< \_CharT, \_Traits >, 1256
  - std::basic\_istream< \_CharT, \_Traits >, 1295
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2885
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1334
  - std::basic\_ofstream< \_CharT, \_Traits >, 1369
  - std::basic\_ostream< \_CharT, \_Traits >, 1400
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2913
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1433
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1701
  - std::ios\_base, 2262, 2263
- predefined\_ops.h, 3430
- prefix
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2415
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, 3115, 3116
- prefix\_search\_node\_update\_imp.hpp, 3650
- prev\_permutation
  - Sorting, 78
- print, 3747
- priority\_queue
  - \_\_gnu\_pbds::priority\_queue< \_Tv, Cmp\_Fn, Tag, \_Alloc >, 2764
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, 2766
- priority\_queue.hpp, 3654
- priority\_queue\_base\_dispatch.hpp, 3645
- probabilities
  - std::discrete\_distribution< \_IntType >, 2063
- probe\_fn\_base.hpp, 3637
- propagate\_const, 3603
- propagate\_on\_container\_copy\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 734
  - std::allocator\_traits< \_Alloc >, 1033
  - std::allocator\_traits< allocator< \_Tp > >, 1039
  - std::allocator\_traits< allocator< void > >, 1048

- std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, [1056](#), [1057](#)
- propagate\_on\_container\_move\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [734](#)
  - std::allocator\_traits< \_Alloc >, [1033](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1039](#)
  - std::allocator\_traits< allocator< void > >, [1048](#)
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, [1057](#)
- propagate\_on\_container\_swap
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [735](#)
  - std::allocator\_traits< \_Alloc >, [1033](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1040](#)
  - std::allocator\_traits< allocator< void > >, [1048](#)
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, [1057](#)
- ptr\_fun
  - Adaptors for pointers to functions, [309](#)
- ptr\_traits.h, [3431](#)
- pubimbue
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2092](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [2998](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3015](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1100](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1464](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1659](#)
  - std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, [3335](#)
- pubseekoff
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2092](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [2998](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3016](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1100](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1465](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1659](#)
  - std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, [3335](#)
- pubseekpos
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2093](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [2999](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3016](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1100](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1465](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1659](#)
  - std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, [3336](#)
- pubsetbuf
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2093](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [2999](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3016](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1101](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1465](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1660](#)
- std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, [3336](#)
- pubsync
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2093](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [2999](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3017](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1101](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1465](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1660](#)
  - std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, [3336](#)
- push
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [2767](#)
  - std::queue< \_Tp, \_Sequence >, [2777](#)
  - std::stack< \_Tp, \_Sequence >, [2986](#)
- push\_back
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [846](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1621](#), [1622](#)
  - std::deque< \_Tp, \_Alloc >, [2048](#)
  - std::list< \_Tp, \_Alloc >, [2341](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2080](#)
  - std::vector< \_Tp, \_Alloc >, [3304](#)
  - std::vector< bool, \_Alloc >, [3324](#)
- push\_front
  - \_\_gnu\_parallel::RestrictedBoundedConcurrentQueue< \_Tp >, [956](#)
  - std::deque< \_Tp, \_Alloc >, [2049](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2154](#)
  - std::list< \_Tp, \_Alloc >, [2341](#)
- push\_heap
  - Heap, [90](#), [91](#)
- put
  - std::basic\_fstream< \_CharT, \_Traits >, [1144](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1256](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1370](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1401](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2915](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1433](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1701](#)
  - std::money\_put< \_CharT, \_Outlter >, [2447](#), [2448](#)
  - std::num\_put< \_CharT, \_Outlter >, [2566](#)–[2571](#)
  - std::time\_put< \_CharT, \_Outlter >, [3086](#), [3087](#)
  - std::time\_put\_byname< \_CharT, \_Outlter >, [3089](#), [3090](#)
- put\_money
  - std, [633](#)
- put\_time
  - std, [633](#)
- putback
  - std::basic\_fstream< \_CharT, \_Traits >, [1145](#)



- std::basic\_ifstream< \_CharT, \_Traits >, 1189
- std::basic\_iostream< \_CharT, \_Traits >, 1257
- std::basic\_istream< \_CharT, \_Traits >, 1295
- std::basic\_istream< \_CharT, \_Traits >::sentry, 2885
- std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1334
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1702
- pword
  - std::basic\_fstream< \_CharT, \_Traits >, 1145
  - std::basic\_ifstream< \_CharT, \_Traits >, 1189
  - std::basic\_ios< \_CharT, \_Traits >, 1215
  - std::basic\_iostream< \_CharT, \_Traits >, 1257
  - std::basic\_istream< \_CharT, \_Traits >, 1296
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2886
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1335
  - std::basic\_ofstream< \_CharT, \_Traits >, 1370
  - std::basic\_ostream< \_CharT, \_Traits >, 1401
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2915
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1434
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1702
  - std::ios\_base, 2263
- qsb\_steals
  - \_\_gnu\_parallel::Settings, 1013
- quadratic\_probe\_fn\_imp.hpp, 3637
- queue, 3747
  - std::queue< \_Tp, \_Sequence >, 2776
- queue.h, 3741
  - \_GLIBCXX\_VOLATILE, 3741
- quicksort.h, 3741
- quiet\_NaN
  - std::numeric\_limits< \_Tp >, 2574
  - std::numeric\_limits< bool >, 2579
  - std::numeric\_limits< char >, 2585
  - std::numeric\_limits< char16\_t >, 2590
  - std::numeric\_limits< char32\_t >, 2595
  - std::numeric\_limits< double >, 2600
  - std::numeric\_limits< float >, 2606
  - std::numeric\_limits< int >, 2611
  - std::numeric\_limits< long >, 2616
  - std::numeric\_limits< long double >, 2621
  - std::numeric\_limits< long long >, 2627
  - std::numeric\_limits< short >, 2632
  - std::numeric\_limits< signed char >, 2637
  - std::numeric\_limits< unsigned char >, 2642
  - std::numeric\_limits< unsigned int >, 2648
  - std::numeric\_limits< unsigned long >, 2653
  - std::numeric\_limits< unsigned long long >, 2658
  - std::numeric\_limits< unsigned short >, 2663
  - std::numeric\_limits< wchar\_t >, 2669
- quoted\_string.h, 3432
- r\_erase\_fn\_imps.hpp, 3612
- radix
  - std::\_\_numeric\_limits\_base, 790
  - std::numeric\_limits< \_Tp >, 2576
  - std::numeric\_limits< bool >, 2582
  - std::numeric\_limits< char >, 2587
  - std::numeric\_limits< char16\_t >, 2592
  - std::numeric\_limits< char32\_t >, 2597
  - std::numeric\_limits< double >, 2603
  - std::numeric\_limits< float >, 2608
  - std::numeric\_limits< int >, 2613
  - std::numeric\_limits< long >, 2618
  - std::numeric\_limits< long double >, 2624
  - std::numeric\_limits< long long >, 2629
  - std::numeric\_limits< short >, 2634
  - std::numeric\_limits< signed char >, 2639
  - std::numeric\_limits< unsigned char >, 2645
  - std::numeric\_limits< unsigned int >, 2650
  - std::numeric\_limits< unsigned long >, 2655
  - std::numeric\_limits< unsigned long long >, 2660
  - std::numeric\_limits< unsigned short >, 2666
  - std::numeric\_limits< wchar\_t >, 2671
- random, 3748
- Random Number Distributions, 234
- Random Number Generation, 233
  - generate\_canonical, 233
- Random Number Generators, 244
  - minstd\_rand, 245
  - minstd\_rand0, 245
  - mt19937, 245
  - mt19937\_64, 245
  - operator<<, 246
- Random Number Utilities, 246
- random.h, 3432
- random.tcc, 3435, 3438
- random\_number.h, 3741
- random\_sample
  - SGL, 155, 156
- random\_sample\_n
  - SGL, 156
- random\_shuffle
  - Mutating, 29, 30
- random\_shuffle.h, 3742
- random\_shuffle\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1013
- range\_access.h, 3440
- ranged\_hash\_fn.hpp, 3637
- ranged\_probe\_fn.hpp, 3637
- ranges, 3748
- ranges\_algo.h, 3442
- ranges\_algobase.h, 3447
- ranges\_base.h, 3447

- ranges\_cmp.h, [3447](#)
- ranges\_uninitialized.h, [3447](#)
- ranges\_util.h, [3447](#)
- ratio, [3748–3750](#)
- ratio\_add
  - Rational Arithmetic, [355](#)
- ratio\_divide
  - Rational Arithmetic, [355](#)
- ratio\_multiply
  - Rational Arithmetic, [355](#)
- ratio\_subtract
  - Rational Arithmetic, [356](#)
- Rational Arithmetic, [354](#)
  - ratio\_add, [355](#)
  - ratio\_divide, [355](#)
  - ratio\_multiply, [355](#)
  - ratio\_subtract, [356](#)
- rb\_tree, [3659](#)
- rb\_tree\_hpp, [3645](#)
- rbegin
  - gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>, [846](#)
  - std, [635](#)
  - std::basic\_string<\_CharT, \_Traits, \_Alloc>, [1622](#)
  - std::deque<\_Tp, \_Alloc>, [2049](#)
  - std::list<\_Tp, \_Alloc>, [2342](#)
  - std::map<\_Key, \_Tp, \_Compare, \_Alloc>, [2401](#), [2402](#)
  - std::multimap<\_Key, \_Tp, \_Compare, \_Alloc>, [2495](#)
  - std::multiset<\_Key, \_Compare, \_Alloc>, [2520](#)
  - std::set<\_Key, \_Compare, \_Alloc>, [2948](#)
  - std::vector<\_Tp, \_Alloc>, [3304](#)
  - std::vector<bool, \_Alloc>, [3324](#), [3325](#)
- rc.hpp, [3645](#)
- rc\_binomial\_heap\_hpp, [3646](#)
- rc\_string\_base.h, [3660](#)
- rdbuf
  - std::basic\_fstream<\_CharT, \_Traits>, [1146](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [1190](#)
  - std::basic\_ios<\_CharT, \_Traits>, [1216](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [1258](#)
  - std::basic\_istream<\_CharT, \_Traits>, [1296](#), [1297](#)
  - std::basic\_istream<\_CharT, \_Traits>::sentry, [2886](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1335](#), [1336](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [1370](#), [1371](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [1401](#), [1402](#)
  - std::basic\_ostream<\_CharT, \_Traits>::sentry, [2915](#)
  - std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1434](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1703](#)
- rdstate
- std::basic\_fstream<\_CharT, \_Traits>, [1146](#)
- std::basic\_ifstream<\_CharT, \_Traits>, [1190](#)
- std::basic\_ios<\_CharT, \_Traits>, [1217](#)
- std::basic\_iostream<\_CharT, \_Traits>, [1258](#)
- std::basic\_istream<\_CharT, \_Traits>, [1297](#)
- std::basic\_istream<\_CharT, \_Traits>::sentry, [2887](#)
- std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1336](#)
- std::basic\_ofstream<\_CharT, \_Traits>, [1371](#)
- std::basic\_ostream<\_CharT, \_Traits>, [1402](#)
- std::basic\_ostream<\_CharT, \_Traits>::sentry, [2916](#)
- std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1435](#)
- std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1703](#)
- read
  - std::basic\_fstream<\_CharT, \_Traits>, [1147](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [1191](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [1259](#)
  - std::basic\_istream<\_CharT, \_Traits>, [1297](#)
  - std::basic\_istream<\_CharT, \_Traits>::sentry, [2887](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1336](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1703](#)
- readsome
  - std::basic\_fstream<\_CharT, \_Traits>, [1147](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [1191](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [1259](#)
  - std::basic\_istream<\_CharT, \_Traits>, [1298](#)
  - std::basic\_istream<\_CharT, \_Traits>::sentry, [2887](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1337](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1704](#)
- ready
  - std::match\_results<\_Bi\_iter, \_Alloc>, [2415](#)
- rebind
  - std::pointer\_traits<\_Ptr>, [2755](#)
  - std::pointer\_traits<\_Tp\*>, [2756](#)
- reduce
  - Generalized Numeric operations, [13](#), [14](#)
- ref
  - std::reference\_wrapper<\_Tp>, [2814](#)
- reference
  - gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it<\_Node, Const\_Iterator, Iterator, \_Alloc>, [1724](#)
  - gnu\_pbds::detail::bin\_search\_tree\_node\_it<\_Node, Const\_Iterator, Iterator, \_Alloc>, [1729](#)
  - gnu\_pbds::detail::binary\_heap\_const\_iterator<\_Value\_Type, Entry, Simple, \_Alloc>, [1739](#)
  - gnu\_pbds::detail::binary\_heap\_point\_const\_iterator<\_Value\_Type, Entry, Simple, \_Alloc>, [1742](#)
  - gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator<

- Node, \_Alloc >, 2306
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_iterator< Node, \_Alloc >, 2310
- std::back\_insert\_iterator< \_Container >, 1077
- std::front\_insert\_iterator< \_Container >, 2163
- std::insert\_iterator< \_Container >, 2253
- std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2292
- std::istreambuf\_iterator< \_CharT, \_Traits >, 2295
- std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 2300
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 2687
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 2691
- std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 2796
- std::set< \_Key, \_Compare, \_Alloc >, 2932
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3171
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3202
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3230
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3257
- refwrap.h, 3449
- regex, 3750, 3751
  - Regular Expressions, 255
- regex.h, 3449
- regex.tcc, 3451
- regex\_automaton.h, 3452
- regex\_automaton.tcc, 3452
- regex\_compiler.h, 3452
- regex\_compiler.tcc, 3453
- regex\_constants.h, 3453
- regex\_error
  - std::regex\_error, 2815
- regex\_error.h, 3455
- regex\_executor.h, 3455
- regex\_executor.tcc, 3456
- regex\_iterator
  - std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 2817
- regex\_match
  - Regular Expressions, 259–262
- regex\_replace
  - Regular Expressions, 263–266
- regex\_scanner.h, 3456
- regex\_scanner.tcc, 3456
- regex\_search
  - Regular Expressions, 266, 268–270
- regex\_token\_iterator
  - std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 2819–2821
- regex\_traits
  - std::regex\_traits< \_Ch\_type >, 2824
  - register\_callback\_iterator< \_Tp, \_CharT, \_Traits >, 1148
  - std::basic\_fstream< \_CharT, \_Traits >, 1148
  - std::basic\_ifstream< \_CharT, \_Traits >, 1192
  - std::basic\_ios< \_CharT, \_Traits >, 1217
  - std::basic\_iostream< \_CharT, \_Traits >, 1260
  - std::basic\_istream< \_CharT, \_Traits >, 1299
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2888
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1337
  - std::basic\_ofstream< \_CharT, \_Traits >, 1371
  - std::basic\_ostream< \_CharT, \_Traits >, 1403
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2916
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1435
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1705
  - std::ios\_base, 2263
  - Regular Expressions, 252
  - cregex\_token\_iterator, 255
  - csub\_match, 255
  - operator<<, 256
  - operator<=>, 256, 257
  - operator==, 257, 258
  - regex, 255
  - regex\_match, 259–262
  - regex\_replace, 263–266
  - regex\_search, 266, 268–270
  - sregex\_token\_iterator, 255
  - ssub\_match, 255
  - swap, 271
  - wcregex\_token\_iterator, 255
  - wcsub\_match, 255
  - wregex, 255
  - wsregex\_token\_iterator, 255
  - wssub\_match, 255
- rehash
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3192
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3221
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3248
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3276
- reinterpret\_pointer\_cast
  - Pointer Abstractions, 341
- release
  - std::auto\_ptr< \_Tp >, 1075
  - std::unique\_ptr< \_Tp, \_Dp >, 3153
  - std::unique\_ptr< \_Tp[], \_Dp >, 3161
- remove
  - Mutating, 30
  - std::forward\_list< \_Tp, \_Alloc >, 2154

- std::list< \_Tp, \_Alloc >, 2342
- remove\_all\_extents\_t
  - Metaprogramming, 351
- remove\_copy
  - Mutating, 31
- remove\_copy\_if
  - Mutating, 31
- remove\_extent\_t
  - Metaprogramming, 352
- remove\_if
  - Mutating, 32
  - std::forward\_list< \_Tp, \_Alloc >, 2155
  - std::list< \_Tp, \_Alloc >, 2342
- remove\_pointer\_t
  - Metaprogramming, 352
- remove\_reference\_t
  - Metaprogramming, 352
- rend
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 847
  - std, 636
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1623
  - std::deque< \_Tp, \_Alloc >, 2049
  - std::list< \_Tp, \_Alloc >, 2343
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2402
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2496
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2521
  - std::set< \_Key, \_Compare, \_Alloc >, 2949
  - std::vector< \_Tp, \_Alloc >, 3305
  - std::vector< bool, \_Alloc >, 3325
- replace
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 847–853
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1527–1536
  - Mutating, 32
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1623–1635
- replace\_copy
  - std, 636
- replace\_copy\_if
  - Mutating, 33
- replace\_if
  - Mutating, 33
- replace\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1014
- requires\_hosted.h, 3456
- reserve
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 853
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1537, 1538
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1636, 1637
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3192
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3222
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3248
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3276
- std::vector< \_Tp, \_Alloc >, 3305
- std::vector< bool, \_Alloc >, 3325
- reset
  - std::auto\_ptr< \_Tp >, 1075
  - std::bernoulli\_distribution, 1719
  - std::binomial\_distribution< \_IntType >, 1750
  - std::bitset< \_Nb >, 1768
  - std::cauchy\_distribution< \_RealType >, 1776
  - std::chi\_squared\_distribution< \_RealType >, 1799
  - std::discrete\_distribution< \_IntType >, 2063
  - std::exponential\_distribution< \_RealType >, 2122
  - std::extreme\_value\_distribution< \_RealType >, 2126
  - std::fisher\_f\_distribution< \_RealType >, 2134
  - std::gamma\_distribution< \_RealType >, 2182
  - std::geometric\_distribution< \_IntType >, 2185
  - std::lognormal\_distribution< \_RealType >, 2367
  - std::negative\_binomial\_distribution< \_IntType >, 2531
  - std::normal\_distribution< \_RealType >, 2537
  - std::piecewise\_constant\_distribution< \_RealType >, 2745
  - std::piecewise\_linear\_distribution< \_RealType >, 2748
  - std::poisson\_distribution< \_IntType >, 2760
  - std::student\_t\_distribution< \_RealType >, 3032
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2080
  - std::uniform\_int\_distribution< \_IntType >, 3144
  - std::uniform\_real\_distribution< \_RealType >, 3147
  - std::unique\_ptr< \_Tp, \_Dp >, 3153
  - std::unique\_ptr< \_Tp[], \_Dp >, 3161, 3162
  - std::weibull\_distribution< \_RealType >, 3347
- resetiosflags
  - std, 637
- resize
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 854
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1539
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, 2237
- Numeric Arrays, 231
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1637, 1638

- std::deque< \_Tp, \_Alloc >, 2049, 2050
- std::forward\_list< \_Tp, \_Alloc >, 2155
- std::list< \_Tp, \_Alloc >, 2343
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2080
- std::vector< \_Tp, \_Alloc >, 3305, 3306
- std::vector< bool, \_Alloc >, 3325, 3326
- resize\_fn\_imps.hpp, 3632
- resize\_no\_store\_hash\_fn\_imps.hpp, 3632
- resize\_policy.hpp, 3625
- resize\_store\_hash\_fn\_imps.hpp, 3632, 3633
- result\_of\_t
  - Metaprogramming, 352
- result\_type
  - \_\_gnu\_cxx::binary\_compose< \_Operation1, \_Operation2, \_Operation3 >, 1733
  - \_\_gnu\_cxx::equal\_to< \_Tp >, 2108
  - \_\_gnu\_cxx::project1st< \_Arg1, \_Arg2 >, 2769
  - \_\_gnu\_cxx::project2nd< \_Arg1, \_Arg2 >, 2770
  - \_\_gnu\_cxx::select1st< \_Pair >, 2861
  - \_\_gnu\_cxx::select2nd< \_Pair >, 2861
  - \_\_gnu\_cxx::subtractive\_rng, 3043
  - \_\_gnu\_cxx::unary\_compose< \_Operation1, \_Operation2 >, 3137
  - \_\_gnu\_parallel::EqualFromLess< \_T1, \_T2, \_Compare >, 876
  - \_\_gnu\_parallel::EqualTo< \_T1, \_T2 >, 877
  - \_\_gnu\_parallel::Less< \_T1, \_T2 >, 903
  - \_\_gnu\_parallel::Lexicographic< \_T1, \_T2, \_Compare >, 904
  - \_\_gnu\_parallel::LexicographicReverse< \_T1, \_T2, \_Compare >, 905
  - \_\_gnu\_parallel::Multiplies< \_Tp1, \_Tp2, \_Result >, 931
  - \_\_gnu\_parallel::Plus< \_Tp1, \_Tp2, \_Result >, 943
  - \_\_gnu\_parallel::\_\_binder1st< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 741
  - \_\_gnu\_parallel::\_\_binder2nd< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 742
  - \_\_gnu\_parallel::\_\_unary\_negate< \_Predicate, argument\_type >, 807
- std::bernoulli\_distribution, 1717
- std::binary\_function< \_Arg1, \_Arg2, \_Result >, 1735
- std::binary\_negate< \_Predicate >, 1745
- std::binder1st< \_Operation >, 1746
- std::binder2nd< \_Operation >, 1748
- std::binomial\_distribution< \_IntType >, 1749
- std::cauchy\_distribution< \_RealType >, 1775
- std::chi\_squared\_distribution< \_RealType >, 1798
- std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 1882
- std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 1883
- std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >, 1884
- std::const\_mem\_fun\_t< \_Ret, \_Tp >, 1885
- std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2057
- std::discrete\_distribution< \_IntType >, 2062
- std::divides< \_Tp >, 2065
- std::divides< void >, 2066
- std::equal\_to< \_Tp >, 2109
- std::equal\_to< void >, 2111
- std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > >, 2703
- std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >, 2705
- std::exponential\_distribution< \_RealType >, 2121
- std::extreme\_value\_distribution< \_RealType >, 2125
- std::fisher\_f\_distribution< \_RealType >, 2133
- std::gamma\_distribution< \_RealType >, 2181
- std::geometric\_distribution< \_IntType >, 2184
- std::greater< \_Tp >, 2197
- std::greater< void >, 2199
- std::greater\_equal< \_Tp >, 2200
- std::greater\_equal< void >, 2202
- std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 2209
- std::hash< \_\_gnu\_cxx::throw\_value\_random >, 2210
- std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2242
- std::less< \_Tp >, 2313
- std::less< void >, 2315
- std::less\_equal< \_Tp >, 2316
- std::less\_equal< void >, 2317
- std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2320
- std::logical\_and< \_Tp >, 2359
- std::logical\_and< void >, 2361
- std::logical\_not< \_Tp >, 2362
- std::logical\_not< void >, 2363
- std::logical\_or< \_Tp >, 2364
- std::logical\_or< void >, 2365
- std::lognormal\_distribution< \_RealType >, 2366
- std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 2419
- std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 2420
- std::mem\_fun\_ref\_t< \_Ret, \_Tp >, 2421
- std::mem\_fun\_t< \_Ret, \_Tp >, 2422
- std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 2425
- std::minus< \_Tp >, 2434
- std::minus< void >, 2435
- std::modulus< \_Tp >, 2437
- std::modulus< void >, 2438
- std::multiplies< \_Tp >, 2499
- std::multiplies< void >, 2500
- std::negate< \_Tp >, 2527
- std::negate< void >, 2528

- `std::negative_binomial_distribution< _IntType >`, 2530
- `std::normal_distribution< _RealType >`, 2536
- `std::not_equal_to< _Tp >`, 2539
- `std::not_equal_to< void >`, 2541
- `std::owner_less< shared_ptr< _Tp > >`, 2704
- `std::owner_less< void >`, 2705
- `std::owner_less< weak_ptr< _Tp > >`, 2706
- `std::piecewise_constant_distribution< _RealType >`, 2743
- `std::piecewise_linear_distribution< _RealType >`, 2747
- `std::plus< _Tp >`, 2750
- `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`, 2752
- `std::pointer_to_unary_function< _Arg, _Result >`, 2754
- `std::poisson_distribution< _IntType >`, 2759
- `std::random_device`, 2781
- `std::seed_seq`, 2860
- `std::shuffle_order_engine< _RandomNumberEngine, __k >`, 2969
- `std::student_t_distribution< _RealType >`, 3031
- `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >`, 3039
- `std::unary_function< _Arg, _Result >`, 3138
- `std::unary_negate< _Predicate >`, 3139
- `std::uniform_int_distribution< _IntType >`, 3143
- `std::uniform_real_distribution< _RealType >`, 3145
- `std::weibull_distribution< _RealType >`, 3346
- `rethrow_exception`
  - `exception_ptr.h`, 3398
  - Exceptions, 130
- `rethrow_if_nested`
  - Exceptions, 130
- `rethrow_nested`
  - `std::nested_exception`, 2532
- `return_temporary_buffer`
  - `std`, 637
- `reverse`
  - Mutating, 34
  - `std::forward_list< _Tp, _Alloc >`, 2156
  - `std::list< _Tp, _Alloc >`, 2343
- `reverse_copy`
  - Mutating, 34
- `reverse_iteration`
  - `__gnu_pbds::container_traits< Cntr >`, 1899
- `reverse_iterator`
  - `std::reverse_iterator< _Iterator >`, 2834
  - `std::set< _Key, _Compare, _Alloc >`, 2932
- `rfind`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 855, 856
- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1539–1543
- `std::basic_string< _CharT, _Traits, _Alloc >`, 1638–1641
- `riemann_zeta`
  - Mathematical Special Functions, 212
  - TR1 Mathematical Special Functions, 251
- `riemann_zetaf`
  - Mathematical Special Functions, 212
- `riemann_zetal`
  - Mathematical Special Functions, 213
- `right`
  - `std`, 637
  - `std::basic_fstream< _CharT, _Traits >`, 1158
  - `std::basic_ifstream< _CharT, _Traits >`, 1201
  - `std::basic_ios< _CharT, _Traits >`, 1224
  - `std::basic_iostream< _CharT, _Traits >`, 1270
  - `std::basic_istream< _CharT, _Traits >`, 1308
  - `std::basic_istream< _CharT, _Traits >::sentry`, 2896
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1347
  - `std::basic_ofstream< _CharT, _Traits >`, 1381
  - `std::basic_ostream< _CharT, _Traits >`, 1411
  - `std::basic_ostream< _CharT, _Traits >::sentry`, 2923
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1445
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1715
  - `std::ios_base`, 2269
- `rope`, 3660
- `ropeimpl.h`, 3663
- `rotate`
  - Mutating, 35
- `rotate_copy`
  - Mutating, 35
- `rotate_fn_imps.hpp`, 3612
- `round`
  - Time, 374, 376
- `round_error`
  - `std::numeric_limits< _Tp >`, 2574
  - `std::numeric_limits< bool >`, 2579
  - `std::numeric_limits< char >`, 2585
  - `std::numeric_limits< char16_t >`, 2590
  - `std::numeric_limits< char32_t >`, 2595
  - `std::numeric_limits< double >`, 2600
  - `std::numeric_limits< float >`, 2606
  - `std::numeric_limits< int >`, 2611
  - `std::numeric_limits< long >`, 2616
  - `std::numeric_limits< long double >`, 2621
  - `std::numeric_limits< long long >`, 2627
  - `std::numeric_limits< short >`, 2632
  - `std::numeric_limits< signed char >`, 2637
  - `std::numeric_limits< unsigned char >`, 2642
  - `std::numeric_limits< unsigned int >`, 2648



- std::numeric\_limits< unsigned long >, 2653
- std::numeric\_limits< unsigned long long >, 2658
- std::numeric\_limits< unsigned short >, 2663
- std::numeric\_limits< wchar\_t >, 2669
- round\_indeterminate
  - std, 581
- round\_style
  - std::\_\_numeric\_limits\_base, 790
  - std::numeric\_limits< \_Tp >, 2576
  - std::numeric\_limits< bool >, 2582
  - std::numeric\_limits< char >, 2587
  - std::numeric\_limits< char16\_t >, 2592
  - std::numeric\_limits< char32\_t >, 2597
  - std::numeric\_limits< double >, 2603
  - std::numeric\_limits< float >, 2608
  - std::numeric\_limits< int >, 2613
  - std::numeric\_limits< long >, 2618
  - std::numeric\_limits< long double >, 2624
  - std::numeric\_limits< long long >, 2629
  - std::numeric\_limits< short >, 2634
  - std::numeric\_limits< signed char >, 2639
  - std::numeric\_limits< unsigned char >, 2645
  - std::numeric\_limits< unsigned int >, 2650
  - std::numeric\_limits< unsigned long >, 2655
  - std::numeric\_limits< unsigned long long >, 2660
  - std::numeric\_limits< unsigned short >, 2666
  - std::numeric\_limits< wchar\_t >, 2671
- round\_to\_nearest
  - std, 581
- round\_toward\_infinity
  - std, 581
- round\_toward\_neg\_infinity
  - std, 581
- round\_toward\_zero
  - std, 581
- runtime\_error
  - std::runtime\_error, 2843
- safe\_base.h, 3585
- safe\_container.h, 3586
- safe\_iterator.h, 3586
- safe\_iterator.tcc, 3587
- safe\_local\_iterator.h, 3588
- safe\_local\_iterator.tcc, 3589
- safe\_sequence.h, 3589
- safe\_sequence.tcc, 3589
- safe\_unordered\_base.h, 3589
- safe\_unordered\_container.h, 3590
- safe\_unordered\_container.tcc, 3590
- sample
  - std, 638
  - std::experimental, 702
- sample\_probe\_fn
  - \_\_gnu\_pbds::sample\_probe\_fn, 2844
- sample\_probe\_fn.hpp, 3638
- sample\_range\_hashing
  - \_\_gnu\_pbds::sample\_range\_hashing, 2845
  - \_\_gnu\_pbds::sample\_resize\_policy, 2849
  - \_\_gnu\_pbds::sample\_resize\_trigger, 2852
  - \_\_gnu\_pbds::sample\_size\_policy, 2853
- sample\_range\_hashing.hpp, 3638
- sample\_ranged\_hash\_fn
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 2846
- sample\_ranged\_hash\_fn.hpp, 3638
- sample\_ranged\_probe\_fn.hpp, 3639
- sample\_resize\_policy
  - \_\_gnu\_pbds::sample\_resize\_policy, 2848
- sample\_resize\_policy.hpp, 3647
- sample\_resize\_trigger
  - \_\_gnu\_pbds::sample\_resize\_trigger, 2850
- sample\_resize\_trigger.hpp, 3647
- sample\_size\_policy
  - \_\_gnu\_pbds::sample\_size\_policy, 2853
- sample\_size\_policy.hpp, 3647
- sample\_tree\_node\_update.hpp, 3650
- sample\_trie\_access\_traits.hpp, 3651
- sample\_trie\_node\_update
  - \_\_gnu\_pbds::sample\_trie\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, 2855
- sample\_trie\_node\_update.hpp, 3651
- sample\_update\_policy
  - \_\_gnu\_pbds::sample\_update\_policy, 2856
- sample\_update\_policy.hpp, 3641
- Sampling Distributions, 242
- sat\_arith.h, 3456
- sbumpc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2093
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2999
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3017
  - std::basic\_filebuf< \_CharT, \_Traits >, 1101
  - std::basic\_streambuf< \_CharT, \_Traits >, 1466
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1660
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3336
- scan\_is
  - std::\_\_ctype\_abstract\_base< \_CharT >, 758
  - std::ctype< \_CharT >, 1919
  - std::ctype< char >, 1947, 1948
  - std::ctype< wchar\_t >, 1973
  - std::ctype\_byname< \_CharT >, 1987
  - std::ctype\_byname< char >, 2006, 2007
- scan\_not
  - std::\_\_ctype\_abstract\_base< \_CharT >, 759
  - std::ctype< \_CharT >, 1920
  - std::ctype< char >, 1948, 1950
  - std::ctype< wchar\_t >, 1973
  - std::ctype\_byname< \_CharT >, 1987
  - std::ctype\_byname< char >, 2007, 2008

## scientific

std, 638  
 std::basic\_fstream< \_CharT, \_Traits >, 1159  
 std::basic\_ifstream< \_CharT, \_Traits >, 1201  
 std::basic\_ios< \_CharT, \_Traits >, 1224  
 std::basic\_iostream< \_CharT, \_Traits >, 1271  
 std::basic\_istream< \_CharT, \_Traits >, 1308  
 std::basic\_istream< \_CharT, \_Traits >::sentry, 2896  
 std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1347  
 std::basic\_ofstream< \_CharT, \_Traits >, 1381  
 std::basic\_ostream< \_CharT, \_Traits >, 1411  
 std::basic\_ostream< \_CharT, \_Traits >::sentry, 2923  
 std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1445  
 std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1716

std::ios\_base, 2269

## scoped\_allocator, 3751

## search

Non-Mutating, 57, 58

std, 638

std::\_\_parallel, 674

## search.h, 3742

## search\_minimal\_n

\_\_gnu\_parallel::Settings, 1014

## search\_n

Non-Mutating, 58, 59

## second

\_\_gnu\_cxx::pair< \_T1, \_T2 >, 2710  
 \_\_gnu\_parallel::\_IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 899  
 std::pair< \_T1, \_T2 >, 2714

## second\_argument\_type

\_\_gnu\_cxx::equal\_to< \_Tp >, 2108  
 \_\_gnu\_cxx::project1st< \_Arg1, \_Arg2 >, 2769  
 \_\_gnu\_cxx::project2nd< \_Arg1, \_Arg2 >, 2770  
 \_\_gnu\_parallel::\_EqualFromLess< \_T1, \_T2, \_Compare >, 876  
 \_\_gnu\_parallel::\_EqualTo< \_T1, \_T2 >, 877  
 \_\_gnu\_parallel::\_Less< \_T1, \_T2 >, 903  
 \_\_gnu\_parallel::\_Lexicographic< \_T1, \_T2, \_Compare >, 904  
 \_\_gnu\_parallel::\_LexicographicReverse< \_T1, \_T2, \_Compare >, 906  
 \_\_gnu\_parallel::\_Multiplies< \_Tp1, \_Tp2, \_Result >, 931  
 \_\_gnu\_parallel::\_Plus< \_Tp1, \_Tp2, \_Result >, 943  
 std::binary\_function< \_Arg1, \_Arg2, \_Result >, 1735  
 std::binary\_negate< \_Predicate >, 1745  
 std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 1882  
 std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 1883  
 std::divides< \_Tp >, 2065

std::divides< void >, 2066

std::equal\_to< \_Tp >, 2109

std::equal\_to< void >, 2111

std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > >, 2703

std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >, 2705

std::greater< \_Tp >, 2197

std::greater< void >, 2199

std::greater\_equal< \_Tp >, 2200

std::greater\_equal< void >, 2202

std::less< \_Tp >, 2313

std::less< void >, 2315

std::less\_equal< \_Tp >, 2316

std::less\_equal< void >, 2317

std::logical\_and< \_Tp >, 2360

std::logical\_and< void >, 2361

std::logical\_or< \_Tp >, 2364

std::logical\_or< void >, 2365

std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 2419

std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 2420

std::minus< \_Tp >, 2434

std::minus< void >, 2435

std::modulus< \_Tp >, 2437

std::modulus< void >, 2438

std::multiplies< \_Tp >, 2499

std::multiplies< void >, 2500

std::not\_equal\_to< \_Tp >, 2540

std::not\_equal\_to< void >, 2541

std::owner\_less< shared\_ptr< \_Tp > >, 2704

std::owner\_less< void >, 2705

std::owner\_less< weak\_ptr< \_Tp > >, 2706

std::plus< \_Tp >, 2750

std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >, 2752

## second\_type

\_\_gnu\_cxx::pair< \_T1, \_T2 >, 2709

\_\_gnu\_parallel::\_IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 896

std::pair< \_T1, \_T2 >, 2712

## seconds

Time, 365

## seed

std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2059

std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2244

std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2321

std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 2971

std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3040

## seed\_seq



- std::seed\_seq, 2860
- seekdir
  - std::basic\_fstream< \_CharT, \_Traits >, 1118
  - std::basic\_ifstream< \_CharT, \_Traits >, 1169
  - std::basic\_ios< \_CharT, \_Traits >, 1208
  - std::basic\_iostream< \_CharT, \_Traits >, 1232
  - std::basic\_istream< \_CharT, \_Traits >, 1278
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2869
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1316
  - std::basic\_ofstream< \_CharT, \_Traits >, 1354
  - std::basic\_ostream< \_CharT, \_Traits >, 1388
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2902
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1420
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1676
  - std::ios\_base, 2260
- seekg
  - std::basic\_fstream< \_CharT, \_Traits >, 1148
  - std::basic\_ifstream< \_CharT, \_Traits >, 1192
  - std::basic\_iostream< \_CharT, \_Traits >, 1260, 1261
  - std::basic\_istream< \_CharT, \_Traits >, 1299
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2888, 2889
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1338
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1705
- seekoff
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2093
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2999
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3017
  - std::basic\_filebuf< \_CharT, \_Traits >, 1101
  - std::basic\_streambuf< \_CharT, \_Traits >, 1466
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1660
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3336
- seekp
  - std::basic\_fstream< \_CharT, \_Traits >, 1149
  - std::basic\_iostream< \_CharT, \_Traits >, 1261
  - std::basic\_ofstream< \_CharT, \_Traits >, 1372
  - std::basic\_ostream< \_CharT, \_Traits >, 1403
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2916, 2917
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1435, 1436
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1706
- seekpos
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2094
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3000
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3017
- std::basic\_filebuf< \_CharT, \_Traits >, 1101
- std::basic\_streambuf< \_CharT, \_Traits >, 1466
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1660
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3337
- select\_on\_container\_copy\_construction
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 738
  - std::allocator\_traits< \_Alloc >, 1036
  - std::allocator\_traits< allocator< \_Tp > >, 1045
  - std::allocator\_traits< allocator< void > >, 1053
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, 1062
- semaphore, 3752
- semaphore\_base.h, 3457
- sentry
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2869
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2903
- Sequences, 121
- sequential
  - \_\_gnu\_parallel, 423
- set, 3752, 3753
  - \_\_gnu\_parallel::Settings, 1010
  - std::bitset< \_Nb >, 1768, 1769
  - std::set< \_Key, \_Compare, \_Alloc >, 2933–2935
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2080, 2081
- Set Operations, 92
  - includes, 93
  - set\_difference, 94
  - set\_intersection, 95, 96
  - set\_symmetric\_difference, 96, 97
  - set\_union, 98
- set.h, 3590
- set\_default\_resource
  - memory\_resource, 3706
- set\_difference
  - Set Operations, 94
- set\_difference\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1014
- set\_intersection
  - Set Operations, 95, 96
- set\_intersection\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1014
- set\_load
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1780
- set\_loads
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2226
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, 2237
- set\_new\_handler
  - std, 638
- set\_num\_threads

- \_\_gnu\_parallel::balanced\_quicksort\_tag, 1086
- \_\_gnu\_parallel::balanced\_tag, 1087
- \_\_gnu\_parallel::default\_parallel\_tag, 2023
- \_\_gnu\_parallel::exact\_tag, 2117
- \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, 2523
- \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, 2524
- \_\_gnu\_parallel::multiway\_mergesort\_tag, 2525
- \_\_gnu\_parallel::omp\_loop\_static\_tag, 2682
- \_\_gnu\_parallel::omp\_loop\_tag, 2683
- \_\_gnu\_parallel::parallel\_tag, 2719
- \_\_gnu\_parallel::quicksort\_tag, 2778
- \_\_gnu\_parallel::sampling\_tag, 2858
- \_\_gnu\_parallel::unbalanced\_tag, 3140
- set\_operations.h, 3743
- set\_symmetric\_difference
  - Set Operations, 96, 97
- set\_symmetric\_difference\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1014
- set\_terminate
  - Exceptions, 130
- set\_unexpected
  - Exceptions, 131
- set\_union
  - Set Operations, 98
- set\_union\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1014
- setbase
  - std, 638
- setbuf
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2094
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3000
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3018
  - std::basic\_filebuf< \_CharT, \_Traits >, 1102
  - std::basic\_streambuf< \_CharT, \_Traits >, 1466
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1661
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3337
- setf
  - std::basic\_fstream< \_CharT, \_Traits >, 1150
  - std::basic\_ifstream< \_CharT, \_Traits >, 1193
  - std::basic\_ios< \_CharT, \_Traits >, 1217, 1218
  - std::basic\_iostream< \_CharT, \_Traits >, 1262
  - std::basic\_istream< \_CharT, \_Traits >, 1300
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2889
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1339
  - std::basic\_ofstream< \_CharT, \_Traits >, 1372, 1373
  - std::basic\_ostream< \_CharT, \_Traits >, 1404
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2917
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1436
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1706, 1707
- std::ios\_base, 2264
- setfill
  - std, 639
- setg
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2094
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3000
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3018
  - std::basic\_filebuf< \_CharT, \_Traits >, 1102
  - std::basic\_streambuf< \_CharT, \_Traits >, 1467
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1661
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3337
- setiosflags
  - std, 639
- setp
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2095
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3001
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3018
  - std::basic\_filebuf< \_CharT, \_Traits >, 1103
  - std::basic\_streambuf< \_CharT, \_Traits >, 1467
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1662
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3338
- setprecision
  - std, 639
- setstate
  - std::basic\_fstream< \_CharT, \_Traits >, 1150
  - std::basic\_ifstream< \_CharT, \_Traits >, 1194
  - std::basic\_ios< \_CharT, \_Traits >, 1218
  - std::basic\_iostream< \_CharT, \_Traits >, 1262
  - std::basic\_istream< \_CharT, \_Traits >, 1300
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2890
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1339
  - std::basic\_ofstream< \_CharT, \_Traits >, 1373
  - std::basic\_ostream< \_CharT, \_Traits >, 1404
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2918
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1437
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1707
- settings.h, 3743
  - \_GLIBCXX\_PARALLEL\_CONDITION, 3744
- setw
  - std, 639
- sgetc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2095
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3001
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3019
  - std::basic\_filebuf< \_CharT, \_Traits >, 1103
  - std::basic\_streambuf< \_CharT, \_Traits >, 1468

- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1662
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3338
- sgetn
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2095
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3001
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3019
- std::basic\_filebuf< \_CharT, \_Traits >, 1103
- std::basic\_streambuf< \_CharT, \_Traits >, 1468
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1662
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3338
- SGL, 147
  - \_Find\_first, 152
  - \_Find\_next, 152
  - \_Unchecked\_flip, 152
  - \_Unchecked\_reset, 152
  - \_Unchecked\_set, 153
  - \_Unchecked\_test, 153
  - \_\_median, 150
  - compose1, 153
  - compose2, 153
  - constant0, 153
  - constant1, 153
  - constant2, 154
  - copy\_n, 154
  - distance, 154
  - identity\_element, 154
  - lexicographical\_compare\_3way, 155
  - power, 155
  - random\_sample, 155, 156
  - random\_sample\_n, 156
  - uninitialized\_copy\_n, 156
- shared\_future
  - std::shared\_future< \_Res >, 2952
  - std::shared\_future< \_Res & >, 2954
  - std::shared\_future< void >, 2956
- shared\_mutex, 3753
- shared\_ptr
  - std::shared\_ptr< \_Tp >, 2960–2966
- shared\_ptr.h, 3457
- shared\_ptr\_atomic.h, 3459
- shared\_ptr\_base.h, 3461
- shift
  - Numeric Arrays, 231
- showbase
  - std, 641
  - std::basic\_fstream< \_CharT, \_Traits >, 1159
  - std::basic\_ifstream< \_CharT, \_Traits >, 1201
  - std::basic\_ios< \_CharT, \_Traits >, 1224
  - std::basic\_iostream< \_CharT, \_Traits >, 1271
  - std::basic\_istream< \_CharT, \_Traits >, 1308
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2896
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1347
  - std::basic\_ofstream< \_CharT, \_Traits >, 1381
  - std::basic\_ostream< \_CharT, \_Traits >, 1411
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2924
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1445
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1716
  - std::ios\_base, 2269
- showmanyc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2096
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3002
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3019
- std::basic\_filebuf< \_CharT, \_Traits >, 1103
- std::basic\_streambuf< \_CharT, \_Traits >, 1468
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1663
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3339
- showpoint
  - std, 641
  - std::basic\_fstream< \_CharT, \_Traits >, 1159
  - std::basic\_ifstream< \_CharT, \_Traits >, 1201
  - std::basic\_ios< \_CharT, \_Traits >, 1224
  - std::basic\_iostream< \_CharT, \_Traits >, 1271
  - std::basic\_istream< \_CharT, \_Traits >, 1308
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2896
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1347
  - std::basic\_ofstream< \_CharT, \_Traits >, 1381
  - std::basic\_ostream< \_CharT, \_Traits >, 1411
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2924
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1445
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1716
  - std::ios\_base, 2269
- showpos
  - std, 641
  - std::basic\_fstream< \_CharT, \_Traits >, 1159
  - std::basic\_ifstream< \_CharT, \_Traits >, 1201
  - std::basic\_ios< \_CharT, \_Traits >, 1224
  - std::basic\_iostream< \_CharT, \_Traits >, 1271
  - std::basic\_istream< \_CharT, \_Traits >, 1308
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2896
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1347
  - std::basic\_ofstream< \_CharT, \_Traits >, 1381
  - std::basic\_ostream< \_CharT, \_Traits >, 1411
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2924
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1445
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1716
  - std::ios\_base, 2269
- shrink\_to\_fit

- `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 856
- `std::basic_string< _CharT, _Traits, _Alloc >`, 1642
- `std::deque< _Tp, _Alloc >`, 2050
- `std::vector< _Tp, _Alloc >`, 3306
- `std::vector< bool, _Alloc >`, 3326
- shuffle
  - Mutating, 36
- shuffle\_order\_engine
  - `std::shuffle_order_engine< _RandomNumberEngine, __k >`, 2969, 2970
- signaling\_NaN
  - `std::numeric_limits< _Tp >`, 2574
  - `std::numeric_limits< bool >`, 2580
  - `std::numeric_limits< char >`, 2585
  - `std::numeric_limits< char16_t >`, 2590
  - `std::numeric_limits< char32_t >`, 2595
  - `std::numeric_limits< double >`, 2601
  - `std::numeric_limits< float >`, 2606
  - `std::numeric_limits< int >`, 2611
  - `std::numeric_limits< long >`, 2616
  - `std::numeric_limits< long double >`, 2622
  - `std::numeric_limits< long long >`, 2627
  - `std::numeric_limits< short >`, 2632
  - `std::numeric_limits< signed char >`, 2637
  - `std::numeric_limits< unsigned char >`, 2643
  - `std::numeric_limits< unsigned int >`, 2648
  - `std::numeric_limits< unsigned long >`, 2653
  - `std::numeric_limits< unsigned long long >`, 2658
  - `std::numeric_limits< unsigned short >`, 2664
  - `std::numeric_limits< wchar_t >`, 2669
- simd, 3604
- `simd_abi::deduce< _Tp, _Np,... >`, 2019
- sin
  - Complex Numbers, 186
  - math.h, 3701
- sinh
  - Complex Numbers, 186
  - math.h, 3701
- size
  - `__gnu_cxx::Temporary_buffer< _ForwardIterator, _Tp >`, 1019
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 856
  - `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`, 3048
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1544
  - Numeric Arrays, 232
  - std, 641
  - `std::Temporary_buffer< _ForwardIterator, _Tp >`, 1021
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1642
  - `std::bitset< _Nb >`, 1769
  - `std::deque< _Tp, _Alloc >`, 2050
  - `std::list< _Tp, _Alloc >`, 2344
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, 2402
  - `std::match_results< _Bi_iter, _Alloc >`, 2415
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 2496
  - `std::multiset< _Key, _Compare, _Alloc >`, 2521
  - `std::priority_queue< _Tp, _Sequence, _Compare >`, 2768
  - `std::queue< _Tp, _Sequence >`, 2777
  - `std::set< _Key, _Compare, _Alloc >`, 2949
  - `std::stack< _Tp, _Sequence >`, 2986
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2081
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3192
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3222
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3248
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3277
  - `std::vector< _Tp, _Alloc >`, 3306
  - `std::vector< bool, _Alloc >`, 3326
- size\_fn\_imps.hpp, 3633
- size\_type
  - `__gnu_pbds::hash_prime_size_policy`, 2232
  - `__gnu_pbds::sample_range_hashing`, 2845
  - `__gnu_pbds::sample_resize_policy`, 2848
  - `__gnu_pbds::sample_resize_trigger`, 2850
  - `__gnu_pbds::sample_size_policy`, 2853
  - `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, 3115
  - `std::allocator_traits< _Alloc >`, 1033
  - `std::allocator_traits< allocator< _Tp > >`, 1040
  - `std::allocator_traits< allocator< void > >`, 1048, 1049
  - `std::allocator_traits< pmr::polymorphic_allocator< _Tp > >`, 1057
  - `std::set< _Key, _Compare, _Alloc >`, 2932
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3172
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3202
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3231
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3257
- skipws
  - std, 641
  - `std::basic_fstream< _CharT, _Traits >`, 1159
  - `std::basic_ifstream< _CharT, _Traits >`, 1201
  - `std::basic_ios< _CharT, _Traits >`, 1224
  - `std::basic_iostream< _CharT, _Traits >`, 1271
  - `std::basic_istream< _CharT, _Traits >`, 1308

- std::basic\_istream< \_CharT, \_Traits >::sentry, [2896](#)
- std::basic\_istream< \_CharT, \_Traits, \_Alloc >, [1347](#)
- std::basic\_ofstream< \_CharT, \_Traits >, [1381](#)
- std::basic\_ostream< \_CharT, \_Traits >, [1412](#)
- std::basic\_ostream< \_CharT, \_Traits >::sentry, [2924](#)
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1445](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1716](#)
- std::ios\_base, [2269](#)
- sleep\_for
  - std::this\_thread, [721](#)
- sleep\_until
  - std::this\_thread, [721](#)
- slice
  - Numeric Arrays, [219](#)
- slice\_array
  - Numeric Arrays, [220](#)
- slice\_array.h, [3462](#)
- slist, [3670](#)
- snnextc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2096](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3002](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3020](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1104](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1469](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1663](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3339](#)
- sort
  - Sorting, [79](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2156](#)
  - std::list< \_Tp, \_Alloc >, [2344](#)
- sort.h, [3744](#)
- sort\_heap
  - Heap, [91](#)
- sort\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1014](#)
- sort\_mwms\_oversampling
  - \_\_gnu\_parallel::Settings, [1014](#)
- sort\_qs\_num\_samples\_preset
  - \_\_gnu\_parallel::Settings, [1014](#)
- sort\_qsb\_base\_case\_maximal\_n
  - \_\_gnu\_parallel::Settings, [1015](#)
- Sorting, [60](#)
  - clamp, [62](#), [63](#)
  - inplace\_merge, [63](#), [64](#)
  - is\_sorted, [64](#), [65](#)
  - is\_sorted\_until, [65](#)
  - lexicographical\_compare, [66](#)
  - lexicographical\_compare\_three\_way, [67](#)
  - max, [67](#), [68](#)
  - max\_element, [68](#), [69](#)
  - merge, [69](#), [70](#)
  - min, [70](#), [71](#)
  - min\_element, [72](#)
  - minmax, [72](#), [73](#)
  - minmax\_element, [73](#)
  - next\_permutation, [74](#)
  - nth\_element, [75](#)
  - partial\_sort, [76](#)
  - partial\_sort\_copy, [77](#)
  - prev\_permutation, [78](#)
  - sort, [79](#)
  - stable\_sort, [80](#)
- source\_location, [3754](#)
- span, [3754](#)
- specfun.h, [3463](#)
- sph\_bessel
  - Mathematical Special Functions, [213](#)
  - TR1 Mathematical Special Functions, [251](#)
- sph\_besself
  - Mathematical Special Functions, [213](#)
- sph\_bessell
  - Mathematical Special Functions, [214](#)
- sph\_legendre
  - Mathematical Special Functions, [214](#)
  - TR1 Mathematical Special Functions, [251](#)
- sph\_legendref
  - Mathematical Special Functions, [214](#)
- sph\_legendrel
  - Mathematical Special Functions, [215](#)
- sph\_neumann
  - Mathematical Special Functions, [215](#)
  - TR1 Mathematical Special Functions, [251](#)
- sph\_neumannf
  - Mathematical Special Functions, [215](#)
- sph\_neumannl
  - Mathematical Special Functions, [215](#)
- splay\_fn\_imps.hpp, [3648](#)
- splay\_tree\_.hpp, [3648](#)
- splice
  - std::list< \_Tp, \_Alloc >, [2344](#), [2345](#)
- splice\_after
  - std::forward\_list< \_Tp, \_Alloc >, [2156](#), [2157](#)
- split\_fn\_imps.hpp, [3644](#)
- split\_join\_can\_throw
  - \_\_gnu\_pbds::container\_traits< Cntr >, [1899](#)
- split\_join\_fn\_imps.hpp, [3626](#)
- sputbackc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2096](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3002](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3020](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1104](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1469](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1663](#)

- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3339
- sputc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2097
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3003
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3020
  - std::basic\_filebuf< \_CharT, \_Traits >, 1104
  - std::basic\_streambuf< \_CharT, \_Traits >, 1469
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1664
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3340
- sputn
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2097
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3003
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3022
  - std::basic\_filebuf< \_CharT, \_Traits >, 1105
  - std::basic\_streambuf< \_CharT, \_Traits >, 1470
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1664
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3340
- sqrt
  - Complex Numbers, 187
  - math.h, 3702
- sregex\_token\_iterator
  - Regular Expressions, 255
- sso\_string\_base.h, 3671
- sstream, 3754
- sstream.tcc, 3465
- ssub\_match
  - Regular Expressions, 255
- stable\_partition
  - Mutating, 36
- stable\_sort
  - Sorting, 80
- stack, 3755
  - std::stack< \_Tp, \_Sequence >, 2986
- standard\_policies.hpp, 3648
- start
  - Numeric Arrays, 232
- state
  - std::fpos< \_StateT >, 2160
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3341
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, 3350
- static\_pointer\_cast
  - Pointer Abstractions, 342
  - std, 642
- std, 464
  - \_Construct, 587
  - \_Destroy, 587
  - \_Destroy\_n, 587
  - \_\_find\_if\_not, 581
  - \_\_find\_if\_not\_n, 581
  - \_\_gcd, 581
  - \_\_gen\_two\_uniform\_ints, 581
  - \_\_inplace\_stable\_sort, 582
  - \_\_lg, 582
  - \_\_merge\_adaptive, 582
  - \_\_merge\_without\_buffer, 583
  - \_\_move\_median\_to\_first, 583
  - \_\_move\_merge, 583
  - \_\_move\_merge\_adaptive, 583
  - \_\_move\_merge\_adaptive\_backward, 584
  - \_\_partition, 584
  - \_\_ptr\_rebind, 578
  - \_\_reverse, 584
  - \_\_rotate, 585
  - \_\_rotate\_adaptive, 585
  - \_\_sample, 585, 586
  - \_\_search\_n\_aux, 586
  - \_\_stable\_partition\_adaptive, 586
  - \_\_umap\_traits, 578
  - \_\_ummap\_traits, 579
  - \_\_umset\_traits, 579
  - \_\_uset\_traits, 579
  - acosh, 587
  - advance, 587
  - begin, 588
  - boolalpha, 589
  - cbegin, 589
  - cend, 589
  - cerr, 646
  - chars\_format, 580
  - cin, 646
  - clog, 646
  - compare\_three\_way\_result\_t, 579
  - const\_pointer\_cast, 589
  - cout, 646
  - crbegin, 589
  - crend, 590
  - data, 590, 591
  - dec, 591
  - default\_sentinel, 646
  - defaultfloat, 591
  - denorm\_absent, 580
  - denorm\_indeterminate, 580
  - denorm\_present, 580
  - destroying\_delete, 646
  - distance, 591
  - dynamic\_pointer\_cast, 592
  - empty, 592
  - end, 592, 593
  - endl, 593
  - ends, 593
  - fabs, 593
  - fixed, 593
  - float\_denorm\_style, 580
  - float\_round\_style, 580
  - flush, 594

- frexp, 594
- from\_chars, 594
- get, 594–596
- get\_money, 596
- get\_new\_handler, 596
- get\_temporary\_buffer, 596
- get\_time, 596
- getline, 597, 598
- hex, 599
- hexfloat, 599
- ignore, 646
- index\_sequence, 579
- index\_sequence\_for, 579
- internal, 599
- io\_errc, 581
- isalnum, 599
- isalpha, 599
- isblank, 600
- iscntrl, 600
- isdigit, 600
- isgraph, 600
- islower, 600
- isprint, 600
- ispunct, 601
- isspace, 601
- isupper, 601
- isxdigit, 601
- left, 601
- make\_index\_sequence, 579
- make\_integer\_sequence, 579
- new\_handler, 579
- noboolalpha, 601
- noshowbase, 602
- noshowpoint, 602
- noshowpos, 602
- noskipws, 602
- nounitbuf, 602
- noupper, 602
- oct, 602
- operator!=, 602, 603
- operator<, 605–607
- operator<=, 608–617
- operator<=, 617, 618
- operator<=>, 618–620
- operator>, 626, 627
- operator>>, 627–632
- operator>=, 627
- operator+, 604, 605
- operator==, 621–626
- operator&, 603
- operator^, 632
- operator|, 633
- put\_money, 633
- put\_time, 633
- rbegin, 635
- rend, 636
- replace\_copy, 636
- resetiosflags, 637
- return\_temporary\_buffer, 637
- right, 637
- round\_indeterminate, 581
- round\_to\_nearest, 581
- round\_toward\_infinity, 581
- round\_toward\_neg\_infinity, 581
- round\_toward\_zero, 581
- sample, 638
- scientific, 638
- search, 638
- set\_new\_handler, 638
- setbase, 638
- setfill, 639
- setiosflags, 639
- setprecision, 639
- setw, 639
- showbase, 641
- showpoint, 641
- showpos, 641
- size, 641
- skipws, 641
- static\_pointer\_cast, 642
- streamoff, 580
- streampos, 580
- streamsize, 580
- swap, 642–644
- tolower, 645
- toupper, 645
- u16streampos, 580
- u32streampos, 580
- unitbuf, 645
- uppercase, 645
- wcerr, 646
- wcin, 646
- wclog, 647
- wcout, 647
- ws, 645
- wstreampos, 580
- std::\_\_basic\_future<\_Res>, 738
- std::\_\_basic\_future<\_Res>::\_M\_get\_result, 740
- std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_StateT>, 742
- std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_StateT>::do\_out, 743
- std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_StateT>::in, 744
- std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_StateT>::out, 744
- std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_StateT>::unshift, 745
- std::\_\_ctype\_abstract\_base<\_CharT>, 749
- std::\_\_ctype\_abstract\_base<\_CharT>::char\_type, 751
- std::\_\_ctype\_abstract\_base<\_CharT>::do\_is, 751
- std::\_\_ctype\_abstract\_base<\_CharT>::do\_narrow, 752



- do\_scan\_is, 753
- do\_scan\_not, 753
- do\_tolower, 754
- do\_toupper, 755
- do\_widen, 756
- is, 757
- narrow, 757, 758
- scan\_is, 758
- scan\_not, 759
- tolower, 759, 760
- toupper, 760
- widen, 761
- std::\_\_cxx11::collate< \_CharT >, 1856
  - ~collate, 1858
  - char\_type, 1857
  - collate, 1857, 1858
  - compare, 1858
  - do\_compare, 1858
  - do\_hash, 1859
  - do\_transform, 1859
  - hash, 1860
  - id, 1861
  - string\_type, 1857
  - transform, 1860
- std::\_\_cxx11::collate\_byname< \_CharT >, 1861
  - char\_type, 1862
  - string\_type, 1862
- std::\_\_debug, 647
  - swap, 654
- std::\_\_debug::bitset< \_Nb >, 1759
- std::\_\_debug::deque< \_Tp, \_Allocator >, 2026
- std::\_\_debug::forward\_list< \_Tp, \_Alloc >, 2137
- std::\_\_debug::list< \_Tp, \_Allocator >, 2324
- std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >, 2376
- std::\_\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator >, 2472
- std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator >, 2500
- std::\_\_debug::set< \_Key, \_Compare, \_Allocator >, 2925
- std::\_\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3163
- std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3194
- std::\_\_debug::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3223
- std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3249
- std::\_\_debug::vector< \_Tp, \_Allocator >, 3285
  - vector, 3288
- std::\_\_detail, 654
  - \_\_from\_chars\_alnum, 656
  - \_\_from\_chars\_pow2\_base, 656
  - operator<<, 656
  - operator>>, 656
- std::\_\_detail::BracketMatcher< \_TraitsT, \_\_icase, \_\_collate >, 864
- std::\_\_detail::Compiler< \_TraitsT >, 867
- std::\_\_detail::Executor< \_Bilter, \_Alloc, \_TraitsT, \_\_dfs\_mode >, 877
- std::\_\_detail::List\_node\_base, 910
- std::\_\_detail::List\_node\_header, 911
- std::\_\_detail::Quoted\_string< \_String, \_CharT >, 951
- std::\_\_detail::Scanner< \_CharT >, 1007
  - \_TokenT, 1008
- std::\_\_detail::StateSeq< \_TraitsT >, 1017
- std::\_\_fdlist::Iterator< \_Const, \_Ptr >, 892
  - operator==, 893
- std::\_\_fdlist::Node< \_ValPtr >, 931
- std::\_\_fdlist::Node\_base< \_VoidPtr >, 934
- std::\_\_is\_fast\_hash< \_Hash >, 776
- std::\_\_is\_location\_invariant< \_Tp >, 776
- std::\_\_new\_allocator< \_Tp >, 784
- std::\_\_numeric\_limits\_base, 786
  - digits, 788
  - digits10, 788
  - has\_denorm, 788
  - has\_denorm\_loss, 788
  - has\_infinity, 788
  - has\_quiet\_NaN, 789
  - has\_signaling\_NaN, 789
  - is\_bounded, 789
  - is\_exact, 789
  - is\_iec559, 789
  - is\_integer, 789
  - is\_modulo, 789
  - is\_signed, 789
  - is\_specialized, 789
  - max\_digits10, 789
  - max\_exponent, 790
  - max\_exponent10, 790
  - min\_exponent, 790
  - min\_exponent10, 790
  - radix, 790
  - round\_style, 790
  - tinyness\_before, 790
  - traps, 790
- std::\_\_parallel, 657
  - search, 674
- std::\_\_parallel::CRandNumber< \_MustBeInt >, 868
- std::\_\_unspecified::\_\_exception\_ptr, 2119
- std::\_Base\_bitset< 0 >, 859
  - \_M\_w, 861
- std::\_Base\_bitset< 1 >, 861
  - \_M\_w, 863
- std::\_Base\_bitset< \_Nw >, 858
  - \_M\_w, 859
- std::\_Bind< \_Signature >, 863



- std::\_Bind\_result< \_Result, \_Signature >, 863
- std::\_Deque\_base< \_Tp, \_Alloc >, 868
  - \_M\_initialize\_map, 869
- std::\_Deque\_iterator< \_Tp, \_Ref, \_Ptr >, 870
  - \_M\_set\_node, 871
- std::\_Function\_base, 879
- std::\_Fwd\_list\_base< \_Tp, \_Alloc >, 880
- std::\_Fwd\_list\_const\_iterator< \_Tp >, 882
  - operator==, 883
- std::\_Fwd\_list\_iterator< \_Tp >, 883
  - operator==, 884
- std::\_Fwd\_list\_node< \_Tp >, 884
- std::\_Fwd\_list\_node\_base, 885
- std::\_List\_base< \_Tp, \_Alloc >, 906
- std::\_List\_const\_iterator< \_Tp >, 907
- std::\_List\_iterator< \_Tp >, 908
- std::\_List\_node< \_Tp >, 909
- std::\_Not\_fn< \_Fn >, 940
- std::\_Placeholder< \_Num >, 942
- std::\_Sp\_ebo\_helper< \_Nm, \_Tp, false >, 1015
- std::\_Sp\_ebo\_helper< \_Nm, \_Tp, true >, 1016
- std::\_Temporary\_buffer< \_ForwardIterator, \_Tp >, 1020
  - \_M\_requested\_size, 1021
  - \_Temporary\_buffer, 1021
  - begin, 1021
  - end, 1021
  - size, 1021
- std::\_Vector\_base< \_Tp, \_Alloc >, 1022
- std::add\_const< \_Tp >, 1023
- std::add\_cv< \_Tp >, 1024
- std::add\_lvalue\_reference< \_Tp >, 1024
- std::add\_pointer< \_Tp >, 1024
- std::add\_rvalue\_reference< \_Tp >, 1024
- std::add\_volatile< \_Tp >, 1025
- std::adopt\_lock\_t, 1025
- std::aligned\_storage< \_Len, \_Align >, 1025
- std::aligned\_union< \_Len, \_Types >, 1026
- std::alignment\_of< \_Tp >, 1026
- std::allocator< \_Tp >, 1029
- std::allocator\_traits< \_Alloc >, 1030
  - allocate, 1034
  - allocator\_type, 1032
  - const\_pointer, 1032
  - const\_void\_pointer, 1032
  - construct, 1034
  - deallocate, 1035
  - destroy, 1035
  - difference\_type, 1033
  - is\_always\_equal, 1033
  - max\_size, 1036
  - pointer, 1033
  - propagate\_on\_container\_copy\_assignment, 1033
  - propagate\_on\_container\_move\_assignment, 1033
  - propagate\_on\_container\_swap, 1033
  - select\_on\_container\_copy\_construction, 1036
  - size\_type, 1033
  - value\_type, 1034
  - void\_pointer, 1034
- std::allocator\_traits< allocator< \_Tp > >, 1036
  - allocate, 1041
  - allocator\_type, 1038
  - const\_pointer, 1038
  - const\_void\_pointer, 1038
  - construct, 1042
  - deallocate, 1043
  - destroy, 1043, 1044
  - difference\_type, 1038, 1039
  - is\_always\_equal, 1039
  - max\_size, 1044
  - pointer, 1039
  - propagate\_on\_container\_copy\_assignment, 1039
  - propagate\_on\_container\_move\_assignment, 1039
  - propagate\_on\_container\_swap, 1040
  - select\_on\_container\_copy\_construction, 1045
  - size\_type, 1040
  - value\_type, 1040
  - void\_pointer, 1040
- std::allocator\_traits< allocator< void > >, 1045
  - allocate, 1049, 1051
  - allocator\_type, 1047
  - const\_pointer, 1047
  - const\_void\_pointer, 1047
  - construct, 1051
  - deallocate, 1051, 1052
  - destroy, 1052
  - difference\_type, 1047
  - is\_always\_equal, 1047, 1048
  - max\_size, 1053
  - pointer, 1048
  - propagate\_on\_container\_copy\_assignment, 1048
  - propagate\_on\_container\_move\_assignment, 1048
  - propagate\_on\_container\_swap, 1048
  - select\_on\_container\_copy\_construction, 1053
  - size\_type, 1048, 1049
  - value\_type, 1049
  - void\_pointer, 1049
- std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, 1054
  - allocate, 1058, 1059
  - allocator\_type, 1055
  - const\_pointer, 1055
  - const\_void\_pointer, 1056
  - construct, 1059, 1060
  - deallocate, 1060
  - destroy, 1061
  - difference\_type, 1056
  - is\_always\_equal, 1056
  - max\_size, 1061, 1062

pointer, 1056  
propagate\_on\_container\_copy\_assignment, 1056, 1057  
propagate\_on\_container\_move\_assignment, 1057  
propagate\_on\_container\_swap, 1057  
select\_on\_container\_copy\_construction, 1062  
size\_type, 1057  
value\_type, 1057, 1058  
void\_pointer, 1058  
std::array<\_Tp, \_Nm >, 1066  
std::atomic<\_Tp >, 1068  
std::atomic<\_Tp \* >, 1069  
std::atomic\_flag, 1071  
std::auto\_ptr<\_Tp >, 1071  
    ~auto\_ptr, 1073  
    auto\_ptr, 1073  
    element\_type, 1072  
    get, 1074  
    operator->, 1074  
    operator=, 1074, 1075  
    operator\*, 1074  
    release, 1075  
    reset, 1075  
std::auto\_ptr\_ref<\_Tp1 >, 1076  
std::back\_insert\_iterator<\_Container >, 1076  
    back\_insert\_iterator, 1078  
    container\_type, 1077  
    iterator\_category, 1077  
    operator++, 1078  
    operator=, 1078  
    operator\*, 1078  
    pointer, 1077  
    reference, 1077  
    value\_type, 1077  
std::bad\_alloc, 1079  
    what, 1079  
std::bad\_cast, 1080  
    what, 1081  
std::bad\_exception, 1081  
    what, 1082  
std::bad\_function\_call, 1082  
    what, 1083  
std::bad\_typeid, 1084  
    what, 1084  
std::bad\_weak\_ptr, 1084  
    what, 1085  
std::basic\_filebuf<\_CharT, \_Traits >, 1089  
    \_M\_buf, 1108  
    \_M\_buf\_locale, 1108  
    \_M\_buf\_size, 1108  
    \_M\_create\_pback, 1093  
    \_M\_destroy\_pback, 1093  
    \_M\_ext\_buf, 1108  
    \_M\_ext\_buf\_size, 1108  
    \_M\_ext\_next, 1108  
    \_M\_in\_beg, 1108  
    \_M\_in\_cur, 1108  
    \_M\_in\_end, 1109  
    \_M\_mode, 1109  
    \_M\_out\_beg, 1109  
    \_M\_out\_cur, 1109  
    \_M\_out\_end, 1109  
    \_M\_pback, 1109  
    \_M\_pback\_cur\_save, 1109  
    \_M\_pback\_end\_save, 1110  
    \_M\_pback\_init, 1110  
    \_M\_reading, 1110  
    \_M\_set\_buffer, 1093  
    ~basic\_filebuf, 1093  
    basic\_filebuf, 1093  
    close, 1094  
    eback, 1094  
    egptr, 1094  
    epptr, 1094  
    gbump, 1095  
    getloc, 1095  
    gptr, 1095  
    imbue, 1096  
    in\_avail, 1096  
    is\_open, 1096  
    open, 1096, 1097  
    overflow, 1098  
    pbackfail, 1098  
    pbase, 1099  
    pbump, 1099  
    pptr, 1099  
    pubimbue, 1100  
    pubseekoff, 1100  
    pubseekpos, 1100  
    pubsetbuf, 1101  
    pubsync, 1101  
    sbumpc, 1101  
    seekoff, 1101  
    seekpos, 1101  
    setbuf, 1102  
    setg, 1102  
    setp, 1103  
    sgetc, 1103  
    sgetn, 1103  
    showmanyc, 1103  
    snextc, 1104  
    sputbackc, 1104  
    sputc, 1104  
    sputn, 1105  
    sungetc, 1105  
    sync, 1105  
    uflow, 1106  
    underflow, 1106

- xsgetn, 1106
- xspn, 1107
- std::basic\_fstream<\_CharT, \_Traits >, 1110
  - \_M\_gcount, 1155
  - \_M\_getloc, 1120
  - ~basic\_fstream, 1119
  - adjustfield, 1155
  - app, 1155
  - ate, 1155
  - bad, 1120
  - badbit, 1155
  - basefield, 1156
  - basic\_fstream, 1118, 1119
  - beg, 1156
  - binary, 1156
  - boolalpha, 1156
  - clear, 1120
  - close, 1120
  - copyfmt, 1120
  - cur, 1156
  - dec, 1156
  - end, 1156
  - eof, 1122
  - eofbit, 1157
  - event, 1118
  - event\_callback, 1117
  - exceptions, 1122
  - fail, 1123
  - failbit, 1157
  - fill, 1123
  - fixed, 1157
  - flags, 1123, 1124
  - floatfield, 1157
  - flush, 1124
  - gcount, 1124
  - get, 1124–1126
  - getline, 1127, 1128
  - getloc, 1128
  - good, 1128
  - goodbit, 1157
  - hex, 1158
  - ignore, 1128, 1129
  - imbue, 1129
  - in, 1158
  - init, 1130
  - internal, 1158
  - iostate, 1117
  - is\_open, 1130
  - isword, 1130
  - left, 1158
  - narrow, 1131
  - oct, 1158
  - open, 1131, 1132
  - openmode, 1118
  - operator bool, 1132
  - operator!, 1132
  - operator<<, 1132–1134, 1136–1138
  - operator>>, 1138–1143
  - out, 1158
  - peek, 1144
  - precision, 1144
  - put, 1144
  - putback, 1145
  - pword, 1145
  - rdbuf, 1146
  - rdstate, 1146
  - read, 1147
  - readsome, 1147
  - register\_callback, 1148
  - right, 1158
  - scientific, 1159
  - seekdir, 1118
  - seekg, 1148
  - seekp, 1149
  - setf, 1150
  - setstate, 1150
  - showbase, 1159
  - showpoint, 1159
  - showpos, 1159
  - skipws, 1159
  - sync, 1151
  - sync\_with\_stdio, 1151
  - tellg, 1151
  - tellp, 1152
  - tie, 1152
  - trunc, 1159
  - unget, 1153
  - unitbuf, 1159
  - unsetf, 1153
  - uppercase, 1159
  - widen, 1153
  - width, 1154
  - write, 1154
  - xalloc, 1155
- std::basic\_ifstream<\_CharT, \_Traits >, 1162
  - \_M\_gcount, 1198
  - \_M\_getloc, 1171
  - \_\_num\_put\_type, 1168
  - ~basic\_ifstream, 1170
  - adjustfield, 1198
  - app, 1198
  - ate, 1198
  - bad, 1171
  - badbit, 1198
  - basefield, 1198
  - basic\_ifstream, 1169, 1170
  - beg, 1198
  - binary, 1198

boolalpha, 1199  
clear, 1171  
close, 1171  
copyfmt, 1171  
cur, 1199  
dec, 1199  
end, 1199  
eof, 1173  
eofbit, 1199  
event, 1169  
event\_callback, 1168  
exceptions, 1173  
fail, 1174  
failbit, 1199  
fill, 1174  
fixed, 1200  
flags, 1174, 1175  
floatfield, 1200  
gcount, 1175  
get, 1175–1177  
getline, 1178, 1179  
getloc, 1179  
good, 1179  
goodbit, 1200  
hex, 1200  
ignore, 1179, 1180  
imbue, 1180  
in, 1200  
init, 1181  
internal, 1200  
iostate, 1168  
is\_open, 1181  
iword, 1181  
left, 1201  
narrow, 1181  
oct, 1201  
open, 1182  
openmode, 1169  
operator bool, 1183  
operator!, 1183  
operator>>, 1183–1188  
out, 1201  
peek, 1188  
precision, 1188, 1189  
putback, 1189  
pword, 1189  
rdbuf, 1190  
rdstate, 1190  
read, 1191  
readsome, 1191  
register\_callback, 1192  
right, 1201  
scientific, 1201  
seekdir, 1169  
seekg, 1192  
setf, 1193  
setstate, 1194  
showbase, 1201  
showpoint, 1201  
showpos, 1201  
skipws, 1201  
sync, 1194  
sync\_with\_stdio, 1194  
tellg, 1195  
tie, 1195  
trunc, 1202  
unget, 1196  
unitbuf, 1202  
unsetf, 1196  
uppercase, 1202  
widen, 1196  
width, 1197  
xalloc, 1197  
std::basic\_ios<\_CharT, \_Traits>, 1203  
    \_M\_getloc, 1210  
    \_\_ctype\_type, 1207  
    \_\_num\_get\_type, 1207  
    \_\_num\_put\_type, 1207  
    ~basic\_ios, 1209  
adjustfield, 1221  
app, 1221  
ate, 1221  
bad, 1210  
badbit, 1221  
basefield, 1221  
basic\_ios, 1209  
beg, 1221  
binary, 1221  
boolalpha, 1222  
char\_type, 1207  
clear, 1210  
copyfmt, 1210  
cur, 1222  
dec, 1222  
end, 1222  
eof, 1211  
eofbit, 1222  
event, 1209  
event\_callback, 1207  
exceptions, 1211  
fail, 1212  
failbit, 1222  
fill, 1212  
fixed, 1223  
flags, 1212, 1213  
floatfield, 1223  
getloc, 1213  
good, 1213

- goodbit, 1223
- hex, 1223
- imbue, 1213
- in, 1223
- init, 1214
- int\_type, 1207
- internal, 1223
- iostate, 1208
- isword, 1214
- left, 1224
- narrow, 1214
- oct, 1224
- off\_type, 1208
- openmode, 1208
- operator bool, 1215
- operator!, 1215
- out, 1224
- pos\_type, 1208
- precision, 1215
- pword, 1215
- rdbuf, 1216
- rdstate, 1217
- register\_callback, 1217
- right, 1224
- scientific, 1224
- seekdir, 1208
- setf, 1217, 1218
- setstate, 1218
- showbase, 1224
- showpoint, 1224
- showpos, 1224
- skipws, 1224
- sync\_with\_stdio, 1218
- tie, 1219
- traits\_type, 1209
- trunc, 1225
- unitbuf, 1225
- unsetf, 1219
- uppercase, 1225
- widen, 1219
- width, 1220
- xalloc, 1220
- std::basic\_ostream<\_CharT, \_Traits >, 1225
  - \_M\_gcount, 1267
  - \_M\_getloc, 1233
  - ~basic\_ostream, 1232
  - adjustfield, 1267
  - app, 1267
  - ate, 1267
  - bad, 1233
  - badbit, 1267
  - basefield, 1268
  - basic\_ostream, 1232
  - beg, 1268
  - binary, 1268
  - boolalpha, 1268
  - clear, 1233
  - copyfmt, 1233
  - cur, 1268
  - dec, 1268
  - end, 1268
  - eof, 1234
  - eofbit, 1269
  - event, 1232
  - event\_callback, 1231
  - exceptions, 1234
  - fail, 1235
  - failbit, 1269
  - fill, 1235
  - fixed, 1269
  - flags, 1235, 1236
  - floatfield, 1269
  - flush, 1236
  - gcount, 1236
  - get, 1236–1238
  - getline, 1239, 1240
  - getloc, 1240
  - good, 1240
  - goodbit, 1269
  - hex, 1270
  - ignore, 1240, 1241
  - imbue, 1241
  - in, 1270
  - init, 1242
  - internal, 1270
  - iostate, 1231
  - isword, 1242
  - left, 1270
  - narrow, 1242
  - oct, 1270
  - openmode, 1231
  - operator bool, 1243
  - operator!, 1243
  - operator<<, 1243, 1244, 1246–1249
  - operator>>, 1249, 1250, 1252–1255
  - out, 1270
  - peek, 1256
  - precision, 1256
  - put, 1256
  - putback, 1257
  - pword, 1257
  - rdbuf, 1258
  - rdstate, 1258
  - read, 1259
  - readsome, 1259
  - register\_callback, 1260
  - right, 1270
  - scientific, 1271

- seekdir, [1232](#)
- seekg, [1260](#), [1261](#)
- seekp, [1261](#)
- setf, [1262](#)
- setstate, [1262](#)
- showbase, [1271](#)
- showpoint, [1271](#)
- showpos, [1271](#)
- skipws, [1271](#)
- sync, [1263](#)
- sync\_with\_stdio, [1263](#)
- tellg, [1263](#)
- tellp, [1264](#)
- tie, [1264](#)
- trunc, [1271](#)
- unget, [1265](#)
- unitbuf, [1271](#)
- unsetf, [1265](#)
- uppercase, [1271](#)
- widen, [1265](#)
- width, [1266](#)
- write, [1266](#)
- xalloc, [1267](#)
- std::basic\_istream< \_CharT, \_Traits >, [1272](#)
  - \_M\_gcount, [1304](#)
  - \_M\_getloc, [1278](#)
  - \_\_num\_put\_type, [1277](#)
  - ~basic\_istream, [1278](#)
  - adjustfield, [1304](#)
  - app, [1304](#)
  - ate, [1304](#)
  - bad, [1279](#)
  - badbit, [1305](#)
  - basefield, [1305](#)
  - basic\_istream, [1278](#)
  - beg, [1305](#)
  - binary, [1305](#)
  - boolalpha, [1305](#)
  - clear, [1279](#)
  - copyfmt, [1279](#)
  - cur, [1305](#)
  - dec, [1305](#)
  - end, [1306](#)
  - eof, [1279](#)
  - eofbit, [1306](#)
  - event, [1278](#)
  - event\_callback, [1277](#)
  - exceptions, [1280](#)
  - fail, [1280](#)
  - failbit, [1306](#)
  - fill, [1281](#)
  - fixed, [1306](#)
  - flags, [1281](#)
  - floatfield, [1306](#)
  - gcount, [1282](#)
  - get, [1282–1284](#)
  - getline, [1284](#), [1285](#)
  - getloc, [1285](#)
  - good, [1286](#)
  - goodbit, [1306](#)
  - hex, [1307](#)
  - ignore, [1286](#)
  - imbue, [1288](#)
  - in, [1307](#)
  - init, [1288](#)
  - internal, [1307](#)
  - iostate, [1277](#)
  - isword, [1288](#)
  - left, [1307](#)
  - narrow, [1289](#)
  - oct, [1307](#)
  - openmode, [1277](#)
  - operator bool, [1289](#)
  - operator!, [1289](#)
  - operator>>, [1289–1294](#)
  - out, [1307](#)
  - peek, [1295](#)
  - precision, [1295](#)
  - putback, [1295](#)
  - pword, [1296](#)
  - rdbuf, [1296](#), [1297](#)
  - rdstate, [1297](#)
  - read, [1297](#)
  - readsome, [1298](#)
  - register\_callback, [1299](#)
  - right, [1308](#)
  - scientific, [1308](#)
  - seekdir, [1278](#)
  - seekg, [1299](#)
  - setf, [1300](#)
  - setstate, [1300](#)
  - showbase, [1308](#)
  - showpoint, [1308](#)
  - showpos, [1308](#)
  - skipws, [1308](#)
  - sync, [1301](#)
  - sync\_with\_stdio, [1301](#)
  - tellg, [1301](#)
  - tie, [1302](#)
  - trunc, [1308](#)
  - unget, [1302](#)
  - unitbuf, [1308](#)
  - unsetf, [1303](#)
  - uppercase, [1308](#)
  - widen, [1303](#)
  - width, [1303](#), [1304](#)
  - xalloc, [1304](#)
- std::basic\_istream< \_CharT, \_Traits >::sentry, [2862](#)

- [\\_M\\_gcount](#), [2893](#)
- [\\_M\\_getloc](#), [2870](#)
- [\\_\\_num\\_put\\_type](#), [2868](#)
- [~basic\\_istream](#), [2870](#)
- [adjustfield](#), [2893](#)
- [app](#), [2893](#)
- [ate](#), [2893](#)
- [bad](#), [2870](#)
- [badbit](#), [2893](#)
- [basefield](#), [2893](#)
- [basic\\_istream](#), [2870](#)
- [beg](#), [2893](#)
- [binary](#), [2893](#)
- [boolalpha](#), [2893](#)
- [clear](#), [2870](#)
- [copyfmt](#), [2870](#)
- [cur](#), [2894](#)
- [dec](#), [2894](#)
- [end](#), [2894](#)
- [eof](#), [2871](#)
- [eofbit](#), [2894](#)
- [event](#), [2869](#)
- [event\\_callback](#), [2868](#)
- [exceptions](#), [2871](#)
- [fail](#), [2872](#)
- [failbit](#), [2894](#)
- [fill](#), [2872](#)
- [fixed](#), [2894](#)
- [flags](#), [2872](#)
- [floatfield](#), [2894](#)
- [fmtflags](#), [2894](#)
- [gcount](#), [2873](#)
- [get](#), [2873](#), [2874](#)
- [getline](#), [2875](#), [2877](#)
- [getloc](#), [2877](#)
- [good](#), [2877](#)
- [goodbit](#), [2895](#)
- [hex](#), [2895](#)
- [ignore](#), [2877](#), [2878](#)
- [imbue](#), [2878](#)
- [in](#), [2895](#)
- [init](#), [2879](#)
- [internal](#), [2895](#)
- [iostate](#), [2868](#)
- [iword](#), [2879](#)
- [left](#), [2895](#)
- [narrow](#), [2879](#)
- [oct](#), [2895](#)
- [openmode](#), [2868](#)
- [operator bool](#), [2880](#)
- [operator!](#), [2880](#)
- [operator>>](#), [2880–2885](#)
- [out](#), [2895](#)
- [peek](#), [2885](#)
- [precision](#), [2885](#)
- [putback](#), [2885](#)
- [pword](#), [2886](#)
- [rdbuf](#), [2886](#)
- [rdstate](#), [2887](#)
- [read](#), [2887](#)
- [readsome](#), [2887](#)
- [register\\_callback](#), [2888](#)
- [right](#), [2896](#)
- [scientific](#), [2896](#)
- [seekdir](#), [2869](#)
- [seekg](#), [2888](#), [2889](#)
- [sentry](#), [2869](#)
- [setf](#), [2889](#)
- [setstate](#), [2890](#)
- [showbase](#), [2896](#)
- [showpoint](#), [2896](#)
- [showpos](#), [2896](#)
- [skipws](#), [2896](#)
- [sync](#), [2890](#)
- [sync\\_with\\_stdio](#), [2890](#)
- [tellg](#), [2890](#)
- [tie](#), [2891](#)
- [traits\\_type](#), [2869](#)
- [trunc](#), [2896](#)
- [unget](#), [2891](#)
- [unitbuf](#), [2896](#)
- [unsetf](#), [2891](#)
- [uppercase](#), [2896](#)
- [widen](#), [2892](#)
- [width](#), [2892](#)
- [xalloc](#), [2892](#)
- [std::basic\\_istream< \\_CharT, \\_Traits, \\_Alloc >](#), [1309](#)
  - [\\_M\\_gcount](#), [1344](#)
  - [\\_M\\_getloc](#), [1318](#)
  - [\\_\\_num\\_put\\_type](#), [1314](#)
  - [~basic\\_istream](#), [1317](#)
  - [adjustfield](#), [1344](#)
  - [app](#), [1344](#)
  - [ate](#), [1344](#)
  - [bad](#), [1318](#)
  - [badbit](#), [1344](#)
  - [basefield](#), [1344](#)
  - [basic\\_istream](#), [1317](#)
  - [beg](#), [1344](#)
  - [binary](#), [1344](#)
  - [boolalpha](#), [1345](#)
  - [clear](#), [1318](#)
  - [copyfmt](#), [1318](#)
  - [cur](#), [1345](#)
  - [dec](#), [1345](#)
  - [end](#), [1345](#)
  - [eof](#), [1319](#)
  - [eofbit](#), [1345](#)

event, 1317  
event\_callback, 1314  
exceptions, 1319  
fail, 1320  
failbit, 1345  
fill, 1320  
fixed, 1346  
flags, 1320, 1321  
floatfield, 1346  
gcount, 1321  
get, 1321–1323  
getline, 1324  
getloc, 1325  
good, 1325  
goodbit, 1346  
hex, 1346  
ignore, 1325  
imbue, 1326  
in, 1346  
init, 1326  
internal, 1346  
iostate, 1316  
iword, 1326  
left, 1347  
narrow, 1327  
oct, 1347  
openmode, 1316  
operator bool, 1327  
operator!, 1327  
operator>>, 1328, 1329, 1331–1333  
out, 1347  
peek, 1334  
precision, 1334  
putback, 1334  
pword, 1335  
rdbuf, 1335, 1336  
rdstate, 1336  
read, 1336  
readsome, 1337  
register\_callback, 1337  
right, 1347  
scientific, 1347  
seekdir, 1316  
seekg, 1338  
setf, 1339  
setstate, 1339  
showbase, 1347  
showpoint, 1347  
showpos, 1347  
skipws, 1347  
str, 1340  
sync, 1340  
sync\_with\_stdio, 1340  
tellg, 1341

tie, 1341  
trunc, 1348  
unget, 1342  
unitbuf, 1348  
unsetf, 1342  
uppercase, 1348  
widen, 1342  
width, 1343  
xalloc, 1343  
std::basic\_ofstream<\_CharT, \_Traits >, 1348  
    \_M\_getloc, 1356  
    \_\_num\_get\_type, 1354  
    ~basic\_ofstream, 1356  
adjustfield, 1378  
app, 1378  
ate, 1378  
bad, 1356  
badbit, 1378  
basefield, 1378  
basic\_ofstream, 1355, 1356  
beg, 1378  
binary, 1378  
boolalpha, 1378  
clear, 1356  
close, 1357  
copyfmt, 1357  
cur, 1379  
dec, 1379  
end, 1379  
eof, 1357  
eofbit, 1379  
event, 1355  
event\_callback, 1354  
exceptions, 1357, 1358  
fail, 1358  
failbit, 1379  
fill, 1358, 1359  
fixed, 1379  
flags, 1359  
floatfield, 1380  
flush, 1359  
getloc, 1360  
good, 1360  
goodbit, 1380  
hex, 1380  
imbue, 1360  
in, 1380  
init, 1361  
internal, 1380  
iostate, 1354  
is\_open, 1361  
iword, 1361  
left, 1380  
narrow, 1361



- oct, [1381](#)
- open, [1363](#)
- openmode, [1354](#)
- operator bool, [1364](#)
- operator!, [1364](#)
- operator<<, [1364–1369](#)
- out, [1381](#)
- precision, [1369](#)
- put, [1370](#)
- pword, [1370](#)
- rdbuf, [1370](#), [1371](#)
- rdstate, [1371](#)
- register\_callback, [1371](#)
- right, [1381](#)
- scientific, [1381](#)
- seekdir, [1354](#)
- seekp, [1372](#)
- setf, [1372](#), [1373](#)
- setstate, [1373](#)
- showbase, [1381](#)
- showpoint, [1381](#)
- showpos, [1381](#)
- skipws, [1381](#)
- sync\_with\_stdio, [1373](#)
- tellp, [1375](#)
- tie, [1375](#)
- trunc, [1381](#)
- unitbuf, [1382](#)
- unsetf, [1375](#)
- uppercase, [1382](#)
- widen, [1376](#)
- width, [1376](#)
- write, [1377](#)
- xalloc, [1377](#)
- std::basic\_ostream< \_CharT, \_Traits >, [1382](#)
  - \_M\_getloc, [1388](#)
  - \_\_num\_get\_type, [1387](#)
  - ~basic\_ostream, [1388](#)
  - adjustfield, [1408](#)
  - app, [1408](#)
  - ate, [1408](#)
  - bad, [1389](#)
  - badbit, [1408](#)
  - basefield, [1408](#)
  - basic\_ostream, [1388](#)
  - beg, [1408](#)
  - binary, [1409](#)
  - boolalpha, [1409](#)
  - clear, [1389](#)
  - copyfmt, [1389](#)
  - cur, [1409](#)
  - dec, [1409](#)
  - end, [1409](#)
  - eof, [1389](#)
  - eofbit, [1409](#)
  - event, [1388](#)
  - event\_callback, [1387](#)
  - exceptions, [1390](#)
  - fail, [1390](#)
  - failbit, [1409](#)
  - fill, [1391](#)
  - fixed, [1410](#)
  - flags, [1391](#)
  - floatfield, [1410](#)
  - flush, [1392](#)
  - getloc, [1392](#)
  - good, [1392](#)
  - goodbit, [1410](#)
  - hex, [1410](#)
  - imbue, [1392](#)
  - in, [1410](#)
  - init, [1393](#)
  - internal, [1411](#)
  - iostate, [1387](#)
  - isword, [1393](#)
  - left, [1411](#)
  - narrow, [1393](#)
  - oct, [1411](#)
  - openmode, [1387](#)
  - operator bool, [1394](#)
  - operator!, [1394](#)
  - operator<<, [1394](#), [1395](#), [1397–1400](#)
  - out, [1411](#)
  - precision, [1400](#)
  - put, [1401](#)
  - pword, [1401](#)
  - rdbuf, [1401](#), [1402](#)
  - rdstate, [1402](#)
  - register\_callback, [1403](#)
  - right, [1411](#)
  - scientific, [1411](#)
  - seekdir, [1388](#)
  - seekp, [1403](#)
  - setf, [1404](#)
  - setstate, [1404](#)
  - showbase, [1411](#)
  - showpoint, [1411](#)
  - showpos, [1411](#)
  - skipws, [1412](#)
  - sync\_with\_stdio, [1405](#)
  - tellp, [1405](#)
  - tie, [1405](#)
  - trunc, [1412](#)
  - unitbuf, [1412](#)
  - unsetf, [1406](#)
  - uppercase, [1412](#)
  - widen, [1406](#)
  - width, [1406](#), [1407](#)

- write, 1407
- xalloc, 1407
- std::basic\_ostream<\_CharT, \_Traits>::sentry, 2897
  - \_M\_getloc, 2903
  - \_\_num\_get\_type, 2901
  - ~basic\_ostream, 2903
  - ~sentry, 2903
  - adjustfield, 2921
  - app, 2921
  - ate, 2921
  - bad, 2903
  - badbit, 2921
  - basefield, 2921
  - basic\_ostream, 2903
  - beg, 2921
  - binary, 2921
  - boolalpha, 2921
  - clear, 2904
  - copyfmt, 2904
  - cur, 2921
  - dec, 2921
  - end, 2921
  - eof, 2904
  - eofbit, 2922
  - event, 2902
  - event\_callback, 2901
  - exceptions, 2904
  - fail, 2905
  - failbit, 2922
  - fill, 2905
  - fixed, 2922
  - flags, 2906
  - floatfield, 2922
  - flush, 2906
  - fmtflags, 2922
  - getloc, 2906
  - good, 2906
  - goodbit, 2923
  - hex, 2923
  - imbue, 2907
  - in, 2923
  - init, 2907
  - internal, 2923
  - iostate, 2901
  - isword, 2907
  - left, 2923
  - narrow, 2907
  - oct, 2923
  - openmode, 2902
  - operator bool, 2908
  - operator!, 2908
  - operator<<, 2908–2913
  - out, 2923
  - precision, 2913
  - put, 2915
  - pword, 2915
  - rdbuf, 2915
  - rdstate, 2916
  - register\_callback, 2916
  - right, 2923
  - scientific, 2923
  - seekdir, 2902
  - seekp, 2916, 2917
  - sentry, 2903
  - setf, 2917
  - setstate, 2918
  - showbase, 2923
  - showpoint, 2924
  - showpos, 2924
  - skipws, 2924
  - sync\_with\_stdio, 2918
  - tellp, 2918
  - tie, 2918
  - trunc, 2924
  - unitbuf, 2924
  - unsetf, 2919
  - uppercase, 2924
  - widen, 2919
  - width, 2919
  - write, 2920
  - xalloc, 2920
- std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc>, 1412
  - \_M\_getloc, 1421
  - \_\_num\_get\_type, 1419
  - ~basic\_ostringstream, 1421
  - adjustfield, 1442
  - app, 1442
  - ate, 1442
  - bad, 1421
  - badbit, 1442
  - basefield, 1442
  - basic\_ostringstream, 1420
  - beg, 1442
  - binary, 1442
  - boolalpha, 1442
  - clear, 1421
  - copyfmt, 1423
  - cur, 1443
  - dec, 1443
  - end, 1443
  - eof, 1423
  - eofbit, 1443
  - event, 1420
  - event\_callback, 1419
  - exceptions, 1423
  - fail, 1424
  - failbit, 1443
  - fill, 1424

- fixed, 1443
- flags, 1425
- floatfield, 1444
- flush, 1425
- getloc, 1425
- good, 1426
- goodbit, 1444
- hex, 1444
- imbue, 1426
- in, 1444
- init, 1426
- internal, 1444
- iostate, 1419
- isword, 1426
- left, 1444
- narrow, 1427
- oct, 1445
- openmode, 1419
- operator bool, 1427
- operator!, 1427
- operator<<, 1428–1432
- out, 1445
- precision, 1433
- put, 1433
- pword, 1434
- rdbuf, 1434
- rdstate, 1435
- register\_callback, 1435
- right, 1445
- scientific, 1445
- seekdir, 1420
- seekp, 1435, 1436
- setf, 1436
- setstate, 1437
- showbase, 1445
- showpoint, 1445
- showpos, 1445
- skipws, 1445
- str, 1437
- sync\_with\_stdio, 1437
- tellp, 1439
- tie, 1439
- trunc, 1445
- unitbuf, 1446
- unsetf, 1439
- uppercase, 1446
- widen, 1440
- width, 1440
- write, 1441
- xalloc, 1441
- std::basic\_regex<\_Ch\_type, \_Rx\_traits >, 1446
  - ~basic\_regex, 1450
  - assign, 1451–1453
  - basic\_regex, 1448–1450
  - flags, 1453
  - getloc, 1453
  - imbue, 1453
  - mark\_count, 1453
  - operator=, 1454
  - swap, 1455
- std::basic\_streambuf<\_CharT, \_Traits >, 1455
  - \_M\_buf\_locale, 1472
  - \_M\_in\_beg, 1472
  - \_M\_in\_cur, 1473
  - \_M\_in\_end, 1473
  - \_M\_out\_beg, 1473
  - \_M\_out\_cur, 1473
  - \_M\_out\_end, 1473
  - \_\_streambuf\_type, 1459
  - ~basic\_streambuf, 1460
  - basic\_streambuf, 1460
  - char\_type, 1459
  - eback, 1460
  - egptr, 1460
  - epptr, 1461
  - gbump, 1461
  - getloc, 1461
  - gptr, 1462
  - imbue, 1462
  - in\_avail, 1462
  - int\_type, 1459
  - off\_type, 1459
  - overflow, 1462
  - pbackfail, 1463
  - pbase, 1463
  - pbump, 1464
  - pos\_type, 1459
  - pptr, 1464
  - pubimbue, 1464
  - pubseekoff, 1465
  - pubseekpos, 1465
  - pubsetbuf, 1465
  - pubsync, 1465
  - sbumpc, 1466
  - seekoff, 1466
  - seekpos, 1466
  - setbuf, 1466
  - setg, 1467
  - setp, 1467
  - sgetc, 1468
  - sgetn, 1468
  - showmanyc, 1468
  - snextc, 1469
  - sputbackc, 1469
  - sputc, 1469
  - sputn, 1470
  - sungetc, 1470
  - sync, 1470

- traits\_type, 1460
- uflow, 1471
- underflow, 1471
- xsggetn, 1471
- xspu, 1472
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1545
  - \_\_resize\_and\_overwrite, 1561
  - ~basic\_string, 1557, 1560
  - append, 1561–1566
  - assign, 1566–1572
  - at, 1572, 1573
  - back, 1574
  - basic\_string, 1554–1560
  - begin, 1574
  - c\_str, 1575
  - capacity, 1575
  - cbegin, 1575
  - cend, 1576
  - clear, 1576
  - compare, 1576–1581
  - copy, 1582
  - crbegin, 1583
  - crend, 1583
  - data, 1583, 1584
  - empty, 1584
  - end, 1584, 1585
  - erase, 1585, 1587, 1588
  - find, 1588–1591
  - find\_first\_not\_of, 1592–1594
  - find\_first\_of, 1595–1597
  - find\_last\_not\_of, 1598–1600
  - find\_last\_of, 1601–1603
  - front, 1604
  - get\_allocator, 1604
  - insert, 1604–1611, 1613, 1614
  - length, 1614, 1615
  - max\_size, 1615
  - npos, 1644
  - operator+=, 1615–1617
  - operator=, 1617–1620
  - operator[], 1620, 1621
  - pop\_back, 1621
  - push\_back, 1621, 1622
  - rbegin, 1622
  - rend, 1623
  - replace, 1623–1635
  - reserve, 1636, 1637
  - resize, 1637, 1638
  - rfind, 1638–1641
  - shrink\_to\_fit, 1642
  - size, 1642
  - substr, 1642, 1643
  - swap, 1643
- std::basic\_string\_view< \_CharT, \_Traits >, 1644
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1650
  - \_M\_buf\_locale, 1667
  - \_M\_in\_beg, 1667
  - \_M\_in\_cur, 1667
  - \_M\_in\_end, 1668
  - \_M\_mode, 1668
  - \_M\_out\_beg, 1668
  - \_M\_out\_cur, 1668
  - \_M\_out\_end, 1668
  - basic\_stringbuf, 1653
  - eback, 1654
  - egptr, 1654
  - epptr, 1654
  - gbump, 1654
  - getloc, 1656
  - gptr, 1656
  - imbue, 1656
  - in\_avail, 1657
  - overflow, 1657
  - pbackfail, 1657
  - pbase, 1658
  - pbump, 1658
  - pptr, 1658
  - pubimbue, 1659
  - pubseekoff, 1659
  - pubseekpos, 1659
  - pubsetbuf, 1660
  - pubsync, 1660
  - sbumpc, 1660
  - seekoff, 1660
  - seekpos, 1660
  - setbuf, 1661
  - setg, 1661
  - setp, 1662
  - sgetc, 1662
  - sgetn, 1662
  - showmanyc, 1663
  - snextc, 1663
  - sputbackc, 1663
  - sputc, 1664
  - sputn, 1664
  - str, 1664, 1665
  - sungetc, 1665
  - sync, 1665
  - uflow, 1665
  - underflow, 1666
  - xsggetn, 1666
  - xspu, 1667
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1668
  - \_M\_gcount, 1712
  - \_M\_getloc, 1678
  - ~basic\_stringstream, 1677
  - adjustfield, 1712
  - app, 1712

ate, 1712  
bad, 1678  
badbit, 1712  
basefield, 1713  
basic\_stringstream, 1677  
beg, 1713  
binary, 1713  
boolalpha, 1713  
clear, 1678  
copyfmt, 1678  
cur, 1713  
dec, 1713  
end, 1713  
eof, 1679  
eofbit, 1714  
event, 1677  
event\_callback, 1675  
exceptions, 1679  
fail, 1680  
failbit, 1714  
fill, 1680  
fixed, 1714  
flags, 1680, 1681  
floatfield, 1714  
flush, 1681  
gcount, 1681  
get, 1681–1683  
getline, 1684, 1685  
getloc, 1685  
good, 1685  
goodbit, 1714  
hex, 1715  
ignore, 1685, 1686  
imbue, 1686  
in, 1715  
init, 1687  
internal, 1715  
iostate, 1676  
iword, 1687  
left, 1715  
narrow, 1687  
oct, 1715  
openmode, 1676  
operator bool, 1688  
operator!, 1688  
operator<<, 1688, 1689, 1691–1694  
operator>>, 1694, 1695, 1697–1700  
out, 1715  
peek, 1701  
precision, 1701  
put, 1701  
putback, 1702  
pword, 1702  
rdbuf, 1703  
rdstate, 1703  
read, 1703  
readsome, 1704  
register\_callback, 1705  
right, 1715  
scientific, 1716  
seekdir, 1676  
seekg, 1705  
seekp, 1706  
setf, 1706, 1707  
setstate, 1707  
showbase, 1716  
showpoint, 1716  
showpos, 1716  
skipws, 1716  
str, 1707, 1708  
sync, 1708  
sync\_with\_stdio, 1708  
tellg, 1709  
tellp, 1709  
tie, 1709  
trunc, 1716  
unget, 1710  
unitbuf, 1716  
unsetf, 1710  
uppercase, 1716  
widen, 1710  
width, 1711  
write, 1711  
xalloc, 1712  
std::bernoulli\_distribution, 1717  
bernoulli\_distribution, 1718  
max, 1718  
min, 1718  
operator(), 1718  
operator==, 1719  
p, 1718  
param, 1718  
reset, 1719  
result\_type, 1717  
std::bernoulli\_distribution::param\_type, 2719  
std::bidirectional\_iterator\_tag, 1719  
std::binary\_function<\_Arg1, \_Arg2, \_Result>, 1733  
first\_argument\_type, 1735  
result\_type, 1735  
second\_argument\_type, 1735  
std::binary\_negate<\_Predicate>, 1744  
first\_argument\_type, 1745  
result\_type, 1745  
second\_argument\_type, 1745  
std::binder1st<\_Operation>, 1745  
argument\_type, 1746  
result\_type, 1746  
std::binder2nd<\_Operation>, 1747

- argument\_type, 1748
- result\_type, 1748
- std::binomial\_distribution< \_IntType >, 1748
  - max, 1749
  - min, 1749
  - operator<<, 1750
  - operator>>, 1752
  - operator(), 1749
  - operator==, 1752
  - p, 1750
  - param, 1750
  - reset, 1750
  - result\_type, 1749
  - t, 1750
- std::binomial\_distribution< \_IntType >::param\_type, 2720
- std::bitset< \_Nb >, 1760
  - all, 1765
  - any, 1765
  - bitset, 1763, 1764
  - count, 1765
  - flip, 1765
  - none, 1765
  - operator<<, 1766
  - operator<=, 1766
  - operator>>, 1766
  - operator>=, 1766
  - operator==, 1766
  - operator&=, 1766
  - operator[], 1767
  - operator~, 1768
  - operator^=, 1767
  - operator|=, 1768
  - reset, 1768
  - set, 1768, 1769
  - size, 1769
  - test, 1769
  - to\_string, 1769
  - to\_ulong, 1770
- std::bitset< \_Nb >::reference, 2812
- std::cauchy\_distribution< \_RealType >, 1774
  - max, 1775
  - min, 1775
  - operator(), 1775
  - operator==, 1776
  - param, 1775
  - reset, 1776
  - result\_type, 1775
- std::cauchy\_distribution< \_RealType >::param\_type, 2720
- std::char\_traits< \_\_gnu\_cxx::character< \_Value, \_Int, \_St > >, 1792
- std::char\_traits< \_CharT >, 1791
- std::char\_traits< wchar\_t >, 1794
- std::chi\_squared\_distribution< \_RealType >, 1797
  - max, 1798
  - min, 1798
  - operator<<, 1799
  - operator>>, 1799
  - operator(), 1798
  - operator==, 1799
  - param, 1798
  - reset, 1799
  - result\_type, 1798
- std::chi\_squared\_distribution< \_RealType >::param\_type, 2721
- std::chrono, 674
  - std::chrono::duration< \_Rep, \_Period >, 2068
  - std::chrono::duration\_values< \_Rep >, 2069
  - std::chrono::gps\_clock, 2196
  - std::chrono::hh\_mm\_ss< \_Duration >, 2238
  - std::chrono::steady\_clock, 3025
  - std::chrono::system\_clock, 3044
  - std::chrono::tai\_clock, 3045
  - std::chrono::time\_point< \_Clock, \_Dur >, 3082
  - std::chrono::treat\_as\_floating\_point< \_Rep >, 3092
  - std::chrono::tzdb\_list, 3135
    - erase\_after, 3136
    - front, 3136
  - std::chrono::tzdb\_list::const\_iterator, 1880
  - std::chrono::utc\_clock, 3279
- std::codecvt< \_InternT, \_ExternT, \_StateT >, 1800
  - do\_always\_noconv, 1801
  - do\_encoding, 1801
  - do\_in, 1802
  - do\_length, 1802
  - do\_max\_length, 1802
  - do\_out, 1802
  - do\_unshift, 1802
  - in, 1803
  - out, 1803
  - unshift, 1804
- std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1805
  - do\_always\_noconv, 1807
  - do\_encoding, 1807
  - do\_in, 1807, 1808
  - do\_length, 1808
  - do\_max\_length, 1808
  - do\_out, 1808, 1809
  - do\_unshift, 1809
  - in, 1809, 1810
  - out, 1811, 1812
  - unshift, 1812, 1813
- std::codecvt< char, char, mbstate\_t >, 1814
  - do\_always\_noconv, 1816
  - do\_encoding, 1816
  - do\_in, 1816
  - do\_length, 1817
  - do\_max\_length, 1817

- do\_out, 1817
- do\_unshift, 1818
- in, 1818, 1819
- out, 1820
- unshift, 1821, 1822
- std::codecvt< char16\_t, char, mbstate\_t >, 1822
  - do\_always\_noconv, 1825
  - do\_encoding, 1825
  - do\_in, 1825
  - do\_length, 1825, 1826
  - do\_max\_length, 1826
  - do\_out, 1826
  - do\_unshift, 1827
  - in, 1827, 1828
  - out, 1828, 1829
  - unshift, 1830
- std::codecvt< char32\_t, char, mbstate\_t >, 1831
  - do\_always\_noconv, 1834
  - do\_encoding, 1834
  - do\_in, 1834
  - do\_length, 1834, 1835
  - do\_max\_length, 1835
  - do\_out, 1835
  - do\_unshift, 1836
  - in, 1836, 1837
  - out, 1837, 1838
  - unshift, 1839
- std::codecvt< wchar\_t, char, mbstate\_t >, 1840
  - do\_always\_noconv, 1843
  - do\_encoding, 1843
  - do\_in, 1843
  - do\_length, 1844
  - do\_max\_length, 1844
  - do\_out, 1844
  - do\_unshift, 1845
  - in, 1845, 1846
  - out, 1847
  - unshift, 1848, 1849
- std::codecvt\_base, 1849
- std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1850
  - do\_always\_noconv, 1852
  - do\_encoding, 1852
  - do\_in, 1852
  - do\_length, 1853
  - do\_max\_length, 1853
  - do\_out, 1853
  - do\_unshift, 1853
  - in, 1854
  - out, 1854
  - unshift, 1855
- std::common\_iterator< \_It, \_Sent >, 1862
- std::common\_type< \_Tp >, 1863
  - std::common\_type< chrono::duration< \_Rep, \_Period > >, 1863
  - std::common\_type< chrono::duration< \_Rep, \_Period >, chrono::duration< \_Rep, \_Period > >, 1864
  - std::common\_type< chrono::duration< \_Rep1, \_Period1 >, chrono::duration< \_Rep2, \_Period2 > >, 1864
  - std::common\_type< chrono::time\_point< \_Clock, \_Duration > >, 1864
  - std::common\_type< chrono::time\_point< \_Clock, \_Duration >, chrono::time\_point< \_Clock, \_Duration > >, 1864
  - std::common\_type< chrono::time\_point< \_Clock, \_Duration1 >, chrono::time\_point< \_Clock, \_Duration2 > >, 1865
  - std::compare\_three\_way\_result< \_Tp, \_Up >, 1865
  - std::complex< \_Tp >, 1865
    - complex, 1867
    - operator+=, 1867
    - operator-=, 1867
    - value\_type, 1866
  - std::complex< double >, 1867
    - complex, 1869
    - operator+=, 1869
    - operator-=, 1869
    - operator/=: 1869, 1870
    - operator=, 1870
    - operator\*=: 1869
    - value\_type, 1868
  - std::complex< float >, 1870
    - complex, 1871
    - operator+=, 1872
    - operator-=, 1872
    - operator/=: 1872
    - operator=, 1872, 1873
    - operator\*=: 1872
    - value\_type, 1871
  - std::complex< long double >, 1873
    - complex, 1874
    - operator+=, 1875
    - operator-=, 1875
    - operator/=: 1875
    - operator=, 1875
    - operator\*=: 1874
    - value\_type, 1874
  - std::condition\_variable, 1877
  - std::condition\_variable\_any, 1878
  - std::conditional< \_Cond, \_Iftrue, \_Iffalse >, 1879
  - std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 1881
    - first\_argument\_type, 1882
    - result\_type, 1882
    - second\_argument\_type, 1882
  - std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 1882
    - first\_argument\_type, 1883

- result\_type, 1883
- second\_argument\_type, 1883
- std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >, 1883
  - argument\_type, 1884
  - result\_type, 1884
- std::const\_mem\_fun\_t< \_Ret, \_Tp >, 1884
  - argument\_type, 1885
  - result\_type, 1885
- std::contiguous\_iterator\_tag, 1903
- std::copyable\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 1904
  - copyable\_function, 1906, 1907
  - operator bool, 1907
  - operator(), 1907
  - operator=, 1907, 1908
  - operator==, 1908
  - swap, 1908
- std::counted\_iterator< \_It >, 1909
- std::ctype< \_CharT >, 1910
  - do\_is, 1912
  - do\_narrow, 1913
  - do\_scan\_is, 1914
  - do\_scan\_not, 1914
  - do\_tolower, 1915
  - do\_toupper, 1916
  - do\_widen, 1917
  - id, 1923
  - is, 1917, 1918
  - narrow, 1918, 1919
  - scan\_is, 1919
  - scan\_not, 1920
  - tolower, 1920, 1921
  - toupper, 1921
  - widen, 1922
- std::ctype< char >, 1923
  - ~ctype, 1927
  - char\_type, 1926
  - classic\_table, 1927
  - ctype, 1926
  - do\_is, 1927, 1928
  - do\_narrow, 1929, 1930, 1932
  - do\_scan\_is, 1933
  - do\_scan\_not, 1934
  - do\_tolower, 1935–1937
  - do\_toupper, 1937–1939
  - do\_widen, 1940, 1941
  - id, 1958
  - is, 1942–1944
  - narrow, 1944–1946
  - scan\_is, 1947, 1948
  - scan\_not, 1948, 1950
  - table, 1951
  - table\_size, 1958
  - tolower, 1951–1953
  - toupper, 1953–1955
  - widen, 1955–1957
- std::ctype< wchar\_t >, 1958
  - ~ctype, 1961
  - char\_type, 1960
  - ctype, 1960, 1961
  - do\_is, 1961, 1962
  - do\_narrow, 1963, 1964
  - do\_scan\_is, 1964, 1965
  - do\_scan\_not, 1965, 1966
  - do\_tolower, 1966–1968
  - do\_toupper, 1968, 1969
  - do\_widen, 1970, 1971
  - id, 1976
  - is, 1971, 1972
  - narrow, 1972, 1973
  - scan\_is, 1973
  - scan\_not, 1973
  - tolower, 1974
  - toupper, 1975
  - widen, 1975, 1976
- std::ctype\_base, 1976
- std::ctype\_byname< \_CharT >, 1977
  - do\_is, 1980
  - do\_narrow, 1980, 1981
  - do\_scan\_is, 1981
  - do\_scan\_not, 1982
  - do\_tolower, 1982, 1983
  - do\_toupper, 1983, 1984
  - do\_widen, 1984
  - id, 1990
  - is, 1985
  - narrow, 1986
  - scan\_is, 1987
  - scan\_not, 1987
  - tolower, 1988
  - toupper, 1988, 1989
  - widen, 1989, 1990
- std::ctype\_byname< char >, 1990
  - classic\_table, 1993
  - do\_is, 1993, 1994
  - do\_narrow, 1995, 1996
  - do\_scan\_is, 1997
  - do\_scan\_not, 1998
  - do\_tolower, 1999, 2000
  - do\_toupper, 2000, 2001
  - do\_widen, 2002, 2003
  - id, 2013
  - is, 2003, 2004
  - narrow, 2005, 2006
  - scan\_is, 2006, 2007
  - scan\_not, 2007, 2008
  - table, 2008
  - table\_size, 2013



- tolower, 2008, 2009
- toupper, 2010, 2011
- widen, 2011, 2012
- std::decay< \_Tp >, 2014
- std::decimal, 681
  - decimal32\_to\_long\_long, 690
- std::decimal::decimal128, 2014
  - decimal128, 2015
- std::decimal::decimal32, 2016
  - decimal32, 2017
- std::decimal::decimal64, 2017
  - decimal64, 2018
- std::default\_delete< \_Tp >, 2019
  - default\_delete, 2020
  - operator(), 2020
- std::default\_delete< \_Tp[] >, 2020
  - default\_delete, 2021
  - operator(), 2021
- std::default\_sentinel\_t, 2024
- std::defer\_lock\_t, 2026
- std::deque< \_Tp, \_Alloc >, 2028
  - \_M\_fill\_initialize, 2036
  - \_M\_new\_elements\_at\_back, 2036
  - \_M\_new\_elements\_at\_front, 2036
  - \_M\_pop\_back\_aux, 2037
  - \_M\_pop\_front\_aux, 2037
  - \_M\_push\_back\_aux, 2037
  - \_M\_push\_front\_aux, 2037
  - \_M\_range\_check, 2037
  - \_M\_range\_initialize, 2037, 2038
  - \_M\_reallocate\_map, 2038
  - \_M\_reserve\_elements\_at\_back, 2038
  - \_M\_reserve\_elements\_at\_front, 2039
  - \_M\_reserve\_map\_at\_back, 2039
  - \_M\_reserve\_map\_at\_front, 2039
  - ~deque, 2036
  - assign, 2039, 2040
  - at, 2040, 2041
  - back, 2041
  - begin, 2041
  - cbegin, 2042
  - cend, 2042
  - clear, 2042
  - crbegin, 2042
  - crend, 2042
  - deque, 2033–2035
  - emplace, 2042
  - empty, 2043
  - end, 2043
  - erase, 2043
  - front, 2044
  - get\_allocator, 2044
  - insert, 2044–2046
  - max\_size, 2046
  - operator=, 2046, 2047
  - operator[], 2047, 2048
  - pop\_back, 2048
  - pop\_front, 2048
  - push\_back, 2048
  - push\_front, 2049
  - rbegin, 2049
  - rend, 2049
  - resize, 2049, 2050
  - shrink\_to\_fit, 2050
  - size, 2050
  - swap, 2050
- std::destroying\_delete\_t, 2051
- std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2056
  - base, 2058
  - discard, 2058
  - discard\_block\_engine, 2057, 2058
  - max, 2059
  - min, 2059
  - operator<<, 2060
  - operator>>, 2060
  - operator(), 2059
  - operator==, 2060
  - result\_type, 2057
  - seed, 2059
- std::discrete\_distribution< \_IntType >, 2061
  - max, 2062
  - min, 2062
  - operator<<, 2063
  - operator>>, 2064
  - operator(), 2062
  - operator==, 2063
  - param, 2062, 2063
  - probabilities, 2063
  - reset, 2063
  - result\_type, 2062
- std::discrete\_distribution< \_IntType >::param\_type, 2721
- std::divides< \_Tp >, 2064
  - first\_argument\_type, 2065
  - result\_type, 2065
  - second\_argument\_type, 2065
- std::divides< void >, 2065
  - first\_argument\_type, 2066
  - result\_type, 2066
  - second\_argument\_type, 2066
- std::domain\_error, 2066
  - what, 2067
- std::enable\_if< bool, \_Tp >, 2083
- std::enable\_shared\_from\_this< \_Tp >, 2083
- std::encoding\_state, 2104
- std::equal\_to< \_Tp >, 2109
  - first\_argument\_type, 2109
  - result\_type, 2109

- second\_argument\_type, 2109
- std::equal\_to< void >, 2110
  - first\_argument\_type, 2111
  - result\_type, 2111
  - second\_argument\_type, 2111
- std::error\_category, 2112
  - default\_error\_condition, 2112
  - equivalent, 2112
  - message, 2112
  - name, 2113
  - operator<= >, 2113
  - operator==, 2113
- std::error\_code, 2113
  - category, 2114
  - default\_error\_condition, 2114
  - error\_code, 2114
  - message, 2114
  - operator bool, 2114
  - value, 2114
- std::error\_condition, 2114
  - assign, 2116
  - category, 2116
  - clear, 2116
  - error\_condition, 2115
  - message, 2116
  - operator bool, 2116
  - value, 2116
- std::exception, 2117
  - what, 2119
- std::exception\_ptr, 2119
- std::experimental, 691
  - gcd, 701
  - get\_deleter, 701
  - is\_bind\_expression\_v, 703
  - is\_placeholder\_v, 703
  - lcm, 701
  - make\_boyer\_moore\_horspool\_searcher, 701
  - make\_boyer\_moore\_searcher, 702
  - make\_default\_searcher, 702
  - make\_ostream\_joiner, 702
  - not\_fn, 702
  - sample, 702
- std::experimental::filesystem::v1::basic\_string\_view< \_CharT, \_Traits >, 1646
- std::experimental::filesystem::v1::filesystem\_error, 2129
  - what, 2130
- std::experimental::filesystem::v1::path, 2735
- std::experimental::filesystem::v1::path::iterator, 2298
- std::experimental::filesystem::v1::space\_info, 2978
- std::experimental::fundamentals\_v1::any, 1064
  - ~any, 1065
  - any, 1065
  - clear, 1065
  - empty, 1065
  - operator=, 1065, 1066
  - swap, 1066
  - type, 1066
- std::experimental::fundamentals\_v1::bad\_any\_cast, 1079
  - what, 1080
- std::experimental::fundamentals\_v1::bad\_optional\_access, 1083
  - what, 1084
- std::experimental::fundamentals\_v1::basic\_string\_view< \_CharT, \_Traits >, 1648
- std::experimental::fundamentals\_v1::in\_place\_t, 2241
- std::experimental::fundamentals\_v1::nullopt\_t, 2543
- std::experimental::fundamentals\_v1::optional< \_Tp >, 2684
- std::experimental::fundamentals\_v2::ostream\_joiner< \_DelimT, \_CharT, \_Traits >, 2688
- std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > >, 2703
  - first\_argument\_type, 2703
  - result\_type, 2703
  - second\_argument\_type, 2703
- std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >, 2705
  - first\_argument\_type, 2705
  - result\_type, 2705
  - second\_argument\_type, 2705
- std::experimental::fundamentals\_v2::propagate\_const< \_Tp >, 2773
- std::exponential\_distribution< \_RealType >, 2120
  - exponential\_distribution, 2121
  - lambda, 2122
  - max, 2122
  - min, 2122
  - operator(), 2122
  - operator==, 2123
  - param, 2122
  - reset, 2122
  - result\_type, 2121
- std::exponential\_distribution< \_RealType >::param\_type, 2722
- std::extent< typename, \_UInt >, 2123
- std::extreme\_value\_distribution< \_RealType >, 2124
  - a, 2125
  - b, 2125
  - max, 2125
  - min, 2125
  - operator(), 2125
  - operator==, 2126
  - param, 2125, 2126
  - reset, 2126
  - result\_type, 2125
- std::extreme\_value\_distribution< \_RealType >::param\_type, 2722
- std::filesystem, 703

std::filesystem::directory\_entry, 2054  
 std::filesystem::directory\_iterator, 2055  
 std::filesystem::file\_status, 2129  
 std::filesystem::filesystem\_error, 2131  
     what, 2131  
 std::filesystem::path, 2738  
     format, 2740  
     operator<<, 2741  
     operator<=>, 2741  
     operator>>, 2741  
     operator/, 2741  
     operator==, 2741  
 std::filesystem::path::iterator, 2298  
 std::filesystem::recursive\_directory\_iterator, 2809  
 std::filesystem::space\_info, 2978  
 std::fisher\_f\_distribution< \_RealType >, 2132  
     max, 2134  
     min, 2134  
     operator<<, 2134  
     operator>>, 2135  
     operator(), 2134  
     operator==, 2135  
     param, 2134  
     reset, 2134  
     result\_type, 2133  
 std::fisher\_f\_distribution< \_RealType >::param\_type, 2723  
 std::forward\_iterator\_tag, 2136  
 std::forward\_list< \_Tp, \_Alloc >, 2140  
     ~forward\_list, 2146  
     assign, 2146, 2147  
     before\_begin, 2147  
     begin, 2148  
     cbefore\_begin, 2148  
     cbegin, 2148  
     cend, 2148  
     clear, 2148  
     emplace\_after, 2148  
     emplace\_front, 2149  
     empty, 2149  
     end, 2149  
     erase\_after, 2150  
     forward\_list, 2143, 2144, 2146  
     front, 2150, 2151  
     get\_allocator, 2151  
     insert\_after, 2151, 2152  
     max\_size, 2152  
     merge, 2153  
     operator=, 2153, 2154  
     pop\_front, 2154  
     push\_front, 2154  
     remove, 2154  
     remove\_if, 2155  
     resize, 2155  
     reverse, 2156  
     sort, 2156  
     splice\_after, 2156, 2157  
     swap, 2157  
     unique, 2158  
 std::fpos< \_StateT >, 2158  
     fpos, 2159  
     operator streamoff, 2159  
     operator+, 2159  
     operator+=, 2159  
     operator-, 2159, 2160  
     operator-=, 2160  
     state, 2160  
 std::from\_chars\_result, 2162  
 std::front\_insert\_iterator< \_Container >, 2162  
     container\_type, 2163  
     front\_insert\_iterator, 2164  
     iterator\_category, 2163  
     operator++, 2164  
     operator=, 2164  
     operator\*, 2164  
     pointer, 2163  
     reference, 2163  
     value\_type, 2163  
 std::function< \_Res(\_ArgTypes...)>, 2165  
     function, 2166, 2167  
     operator bool, 2167  
     operator(), 2167  
     operator=, 2168, 2169  
     swap, 2169  
     target, 2170  
     target\_type, 2170  
 std::function\_ref< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV  
     noexcept(\_Noex)>, 2171  
     function\_ref, 2172  
     operator(), 2173  
 std::future< \_Res >, 2173  
     \_M\_get\_result, 2174  
     future, 2174  
     get, 2174  
 std::future< \_Res & >, 2174  
     \_M\_get\_result, 2176  
     future, 2176  
     get, 2176  
 std::future< void >, 2176  
     \_M\_get\_result, 2178  
     future, 2178  
     get, 2178  
 std::future\_error, 2178  
     what, 2179  
 std::gamma\_distribution< \_RealType >, 2179  
     alpha, 2181  
     beta, 2181  
     gamma\_distribution, 2181

- max, [2181](#)
- min, [2181](#)
- operator<=, [2182](#)
- operator>=, [2183](#)
- operator(), [2181](#), [2182](#)
- operator==, [2183](#)
- param, [2182](#)
- reset, [2182](#)
- result\_type, [2181](#)
- std::gamma\_distribution< \_RealType >::param\_type, [2723](#)
- std::geometric\_distribution< \_IntType >, [2183](#)
  - max, [2184](#)
  - min, [2184](#)
  - operator(), [2185](#)
  - operator==, [2185](#)
  - p, [2185](#)
  - param, [2185](#)
  - reset, [2185](#)
  - result\_type, [2184](#)
- std::geometric\_distribution< \_IntType >::param\_type, [2724](#)
- std::greater< \_Tp >, [2197](#)
  - first\_argument\_type, [2197](#)
  - result\_type, [2197](#)
  - second\_argument\_type, [2197](#)
- std::greater< void >, [2198](#)
  - first\_argument\_type, [2199](#)
  - result\_type, [2199](#)
  - second\_argument\_type, [2199](#)
- std::greater\_equal< \_Tp >, [2199](#)
  - first\_argument\_type, [2200](#)
  - result\_type, [2200](#)
  - second\_argument\_type, [2200](#)
- std::greater\_equal< void >, [2201](#)
  - first\_argument\_type, [2202](#)
  - result\_type, [2202](#)
  - second\_argument\_type, [2202](#)
- std::gslice, [2203](#)
- std::gslice\_array< \_Tp >, [2204](#)
  - operator<=, [2206](#)
  - operator>=, [2206](#)
  - operator+=, [2205](#)
  - operator-=, [2205](#)
  - operator/=, [2205](#)
  - operator%=, [2205](#)
  - operator&=, [2205](#)
  - operator\*=, [2205](#)
  - operator^=, [2206](#)
  - operator|=, [2206](#)
- std::has\_virtual\_destructor< \_Tp >, [2206](#)
- std::hash< \_\_debug::bitset< \_Nb > >, [2207](#)
- std::hash< \_\_debug::vector< bool, \_Alloc > >, [2207](#)
- std::hash< \_\_gnu\_cxx::\_\_u16vstring >, [2207](#)
- std::hash< \_\_gnu\_cxx::\_\_u32vstring >, [2208](#)
- std::hash< \_\_gnu\_cxx::\_\_vstring >, [2208](#)
- std::hash< \_\_gnu\_cxx::\_\_wvstring >, [2208](#)
- std::hash< \_\_gnu\_cxx::throw\_value\_limit >, [2209](#)
  - argument\_type, [2209](#)
  - result\_type, [2209](#)
- std::hash< \_\_gnu\_cxx::throw\_value\_random >, [2210](#)
  - argument\_type, [2210](#)
  - result\_type, [2210](#)
- std::hash< \_\_gnu\_debug::basic\_string< \_CharT > >, [2211](#)
- std::hash< \_\_shared\_ptr< \_Tp, \_Lp > >, [2211](#)
- std::hash< \_Tp >, [2206](#)
- std::hash< \_Tp \* >, [2211](#)
- std::hash< basic\_string< char, char\_traits< char >, \_Alloc > >, [2212](#)
- std::hash< basic\_string< char16\_t, char\_traits< char16\_t >, \_Alloc > >, [2212](#)
- std::hash< basic\_string< char32\_t, char\_traits< char32\_t >, \_Alloc > >, [2212](#)
- std::hash< basic\_string< wchar\_t, char\_traits< wchar\_t >, \_Alloc > >, [2213](#)
- std::hash< bool >, [2213](#)
- std::hash< char >, [2213](#)
- std::hash< char16\_t >, [2214](#)
- std::hash< char32\_t >, [2214](#)
- std::hash< double >, [2214](#)
- std::hash< error\_code >, [2214](#)
- std::hash< error\_condition >, [2215](#)
- std::hash< experimental::optional< \_Tp > >, [2215](#)
- std::hash< experimental::shared\_ptr< \_Tp > >, [2215](#)
- std::hash< float >, [2216](#)
- std::hash< int >, [2216](#)
- std::hash< long >, [2216](#)
- std::hash< long double >, [2216](#)
- std::hash< long long >, [2217](#)
- std::hash< shared\_ptr< \_Tp > >, [2217](#)
- std::hash< short >, [2217](#)
- std::hash< signed char >, [2217](#)
- std::hash< thread::id >, [2218](#)
- std::hash< type\_index >, [2218](#)
- std::hash< unique\_ptr< \_Tp, \_Dp > >, [2218](#)
- std::hash< unsigned char >, [2219](#)
- std::hash< unsigned int >, [2219](#)
- std::hash< unsigned long >, [2219](#)
- std::hash< unsigned long long >, [2219](#)
- std::hash< unsigned short >, [2220](#)
- std::hash< wchar\_t >, [2220](#)
- std::hash<::bitset< \_Nb > >, [2220](#)
- std::hash<::vector< bool, \_Alloc > >, [2220](#)
- std::identity, [2241](#)
- std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, [2241](#)
  - base, [2243](#)

- discard, 2243
- independent\_bits\_engine, 2242, 2243
- max, 2243
- min, 2244
- operator>>, 2245
- operator(), 2244
- operator==, 2244
- result\_type, 2242
- seed, 2244
- std::indirect\_array< \_Tp >, 2245
  - operator<=, 2247
  - operator>=, 2247
  - operator+=, 2247
  - operator-=, 2247
  - operator/=: 2247
  - operator%=: 2247
  - operator&=: 2247
  - operator\*=, 2247
  - operator^=: 2248
  - operator|=, 2248
- std::initializer\_list< \_E >, 2248
  - begin, 2249
  - end, 2249
- std::input\_iterator\_tag, 2249
- std::insert\_iterator< \_Container >, 2251
  - container\_type, 2253
  - insert\_iterator, 2253
  - iterator\_category, 2253
  - operator++, 2253
  - operator=, 2254
  - operator\*, 2253
  - pointer, 2253
  - reference, 2253
  - value\_type, 2253
- std::integer\_sequence< \_Tp, \_Idx >, 2254
- std::integral\_constant< \_Tp, \_\_v >, 2255
- std::invalid\_argument, 2255
  - what, 2256
- std::ios\_base, 2256
  - \_M\_getloc, 2261
  - ~ios\_base, 2261
  - adjustfield, 2266
  - app, 2266
  - ate, 2266
  - badbit, 2266
  - basefield, 2266
  - beg, 2266
  - binary, 2266
  - boolalpha, 2267
  - cur, 2267
  - dec, 2267
  - end, 2267
  - eofbit, 2267
  - event, 2260
  - event\_callback, 2259
  - failbit, 2267
  - fixed, 2268
  - flags, 2261
  - floatfield, 2268
  - getloc, 2261
  - goodbit, 2268
  - hex, 2268
  - imbue, 2262
  - in, 2268
  - internal, 2268
  - iostate, 2260
  - isword, 2262
  - left, 2269
  - oct, 2269
  - openmode, 2260
  - out, 2269
  - precision, 2262, 2263
  - pword, 2263
  - register\_callback, 2263
  - right, 2269
  - scientific, 2269
  - seekdir, 2260
  - setf, 2264
  - showbase, 2269
  - showpoint, 2269
  - showpos, 2269
  - skipws, 2269
  - sync\_with\_stdio, 2264
  - trunc, 2270
  - unitbuf, 2270
  - unsetf, 2265
  - uppercase, 2270
  - width, 2265
  - xalloc, 2265
- std::is\_abstract< \_Tp >, 2270
- std::is\_arithmetic< \_Tp >, 2270
- std::is\_array< \_Tp >, 2270
- std::is\_assignable< \_Tp, \_Up >, 2271
- std::is\_base\_of< \_Base, \_Derived >, 2271
- std::is\_bind\_expression< \_Bind< \_Signature > >, 2272
- std::is\_bind\_expression< \_Bind\_result< \_Result, \_Signature > >, 2272
- std::is\_bind\_expression< \_Tp >, 2271
- std::is\_bind\_expression< const \_Bind< \_Signature > >, 2272
- std::is\_bind\_expression< const \_Bind\_result< \_Result, \_Signature > >, 2272
- std::is\_bind\_expression< const volatile \_Bind< \_Signature > >, 2273
- std::is\_bind\_expression< const volatile \_Bind\_result< \_Result, \_Signature > >, 2273
- std::is\_bind\_expression< volatile \_Bind< \_Signature > >, 2273

- `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >`, 2273
- `std::is_class< _Tp >`, 2274
- `std::is_compound< _Tp >`, 2274
- `std::is_const< _Tp >`, 2274
- `std::is_constructible< _Tp, _Args >`, 2274
- `std::is_copy_assignable< _Tp >`, 2274
- `std::is_copy_constructible< _Tp >`, 2275
- `std::is_default_constructible< _Tp >`, 2275
- `std::is_destructible< _Tp >`, 2275
- `std::is_empty< _Tp >`, 2275
- `std::is_enum< _Tp >`, 2276
- `std::is_error_code_enum< _Tp >`, 2276
- `std::is_error_code_enum< future_errc >`, 2276
- `std::is_error_condition_enum< _Tp >`, 2276
- `std::is_floating_point< _Tp >`, 2277
- `std::is_function< _Tp >`, 2277
- `std::is_fundamental< _Tp >`, 2278
- `std::is_integral< _Tp >`, 2278
- `std::is_layout_compatible< _Tp, _Up >`, 2278
- `std::is_literal_type< _Tp >`, 2278
- `std::is_lvalue_reference< typename >`, 2279
- `std::is_member_function_pointer< _Tp >`, 2279
- `std::is_member_object_pointer< _Tp >`, 2279
- `std::is_member_pointer< _Tp >`, 2279
- `std::is_move_assignable< _Tp >`, 2280
- `std::is_move_constructible< _Tp >`, 2280
- `std::is_nothrow_assignable< _Tp, _Up >`, 2280
- `std::is_nothrow_constructible< _Tp, _Args >`, 2280
- `std::is_nothrow_copy_assignable< _Tp >`, 2280
- `std::is_nothrow_copy_constructible< _Tp >`, 2281
- `std::is_nothrow_default_constructible< _Tp >`, 2281
- `std::is_nothrow_destructible< _Tp >`, 2281
- `std::is_nothrow_move_assignable< _Tp >`, 2281
- `std::is_nothrow_move_constructible< _Tp >`, 2282
- `std::is_object< _Tp >`, 2282
- `std::is_placeholder< _Placeholder< _Num > >`, 2283
- `std::is_placeholder< _Tp >`, 2282
- `std::is_pod< _Tp >`, 2284
- `std::is_pointer< _Tp >`, 2285
- `std::is_pointer_interconvertible_base_of< _Base, _Derived >`, 2285
- `std::is_polymorphic< _Tp >`, 2285
- `std::is_reference< _Tp >`, 2285
- `std::is_rvalue_reference< typename >`, 2286
- `std::is_same< _Tp, _Up >`, 2286
- `std::is_scalar< _Tp >`, 2286
- `std::is_signed< _Tp >`, 2286
- `std::is_standard_layout< _Tp >`, 2287
- `std::is_trivial< _Tp >`, 2287
- `std::is_trivially_assignable< _Tp, _Up >`, 2287
- `std::is_trivially_constructible< _Tp, _Args >`, 2287
- `std::is_trivially_copy_assignable< _Tp >`, 2288
- `std::is_trivially_copy_constructible< _Tp >`, 2288
- `std::is_trivially_copyable< _Tp >`, 2288
- `std::is_trivially_default_constructible< _Tp >`, 2288
- `std::is_trivially_destructible< _Tp >`, 2289
- `std::is_trivially_move_assignable< _Tp >`, 2289
- `std::is_trivially_move_constructible< _Tp >`, 2289
- `std::is_union< _Tp >`, 2289
- `std::is_unsigned< _Tp >`, 2290
- `std::is_void< _Tp >`, 2290
- `std::is_volatile< _Tp >`, 2290
- `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`, 2290
  - `difference_type`, 2292
  - `istream_iterator`, 2292
  - `iterator_category`, 2292
  - `operator==`, 2292
  - `pointer`, 2292
  - `reference`, 2292
  - `value_type`, 2292
- `std::istreambuf_iterator< _CharT, _Traits >`, 2293
  - `char_type`, 2294
  - `difference_type`, 2294
  - `equal`, 2296
  - `int_type`, 2294
  - `istream_type`, 2294
  - `istreambuf_iterator`, 2295, 2296
  - `iterator_category`, 2295
  - `operator++`, 2296
  - `operator*`, 2296
  - `pointer`, 2295
  - `reference`, 2295
  - `streambuf_type`, 2295
  - `traits_type`, 2295
  - `value_type`, 2295
- `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`, 2299
  - `difference_type`, 2300
  - `iterator_category`, 2300
  - `pointer`, 2300
  - `reference`, 2300
  - `value_type`, 2300
- `std::iterator_traits< _Iterator >`, 2301
- `std::iterator_traits< _Tp * >`, 2301
- `std::length_error`, 2311
  - `what`, 2312
- `std::less< _Tp >`, 2312
  - `first_argument_type`, 2313
  - `result_type`, 2313
  - `second_argument_type`, 2313
- `std::less< void >`, 2314
  - `first_argument_type`, 2315
  - `result_type`, 2315
  - `second_argument_type`, 2315
- `std::less_equal< _Tp >`, 2315
  - `first_argument_type`, 2316
  - `result_type`, 2316

- second\_argument\_type, [2316](#)
- std::less\_equal< void >, [2317](#)
  - first\_argument\_type, [2317](#)
  - result\_type, [2317](#)
  - second\_argument\_type, [2317](#)
- std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [2319](#)
  - discard, [2321](#)
  - increment, [2323](#)
  - linear\_congruential\_engine, [2320](#)
  - max, [2321](#)
  - min, [2321](#)
  - modulus, [2323](#)
  - multiplier, [2323](#)
  - operator<<, [2322](#)
  - operator>>, [2322](#)
  - operator(), [2321](#)
  - operator==, [2322](#)
  - result\_type, [2320](#)
  - seed, [2321](#)
- std::list< \_Tp, \_Alloc >, [2327](#)
  - \_M\_create\_node, [2333](#)
  - ~list, [2333](#)
  - assign, [2333](#), [2334](#)
  - back, [2334](#)
  - begin, [2334](#)
  - cbegin, [2334](#)
  - cend, [2335](#)
  - clear, [2335](#)
  - crbegin, [2335](#)
  - crend, [2335](#)
  - emplace, [2335](#)
  - empty, [2335](#)
  - end, [2336](#)
  - erase, [2336](#)
  - front, [2337](#)
  - get\_allocator, [2337](#)
  - insert, [2337](#)–[2339](#)
  - list, [2331](#), [2332](#)
  - max\_size, [2339](#)
  - merge, [2339](#)
  - operator=, [2340](#)
  - pop\_back, [2341](#)
  - pop\_front, [2341](#)
  - push\_back, [2341](#)
  - push\_front, [2341](#)
  - rbegin, [2342](#)
  - remove, [2342](#)
  - remove\_if, [2342](#)
  - rend, [2343](#)
  - resize, [2343](#)
  - reverse, [2343](#)
  - size, [2344](#)
  - sort, [2344](#)
  - splice, [2344](#), [2345](#)
  - swap, [2346](#)
  - unique, [2346](#)
- std::literals, [706](#)
- std::literals::chrono\_literals, [707](#)
- std::locale, [2348](#)
  - ~locale, [2352](#)
  - all, [2356](#)
  - category, [2350](#)
  - classic, [2353](#)
  - collate, [2356](#)
  - combine, [2353](#)
  - ctype, [2356](#)
  - global, [2353](#)
  - has\_facet, [2355](#)
  - locale, [2350](#)–[2352](#)
  - messages, [2356](#)
  - monetary, [2356](#)
  - name, [2354](#)
  - none, [2356](#)
  - numeric, [2357](#)
  - operator(), [2354](#)
  - operator=, [2354](#)
  - operator==, [2354](#)
  - time, [2357](#)
  - use\_facet, [2355](#)
- std::locale::facet, [2126](#)
  - ~facet, [2128](#)
  - facet, [2128](#)
- std::locale::id, [2238](#)
  - has\_facet, [2239](#)
  - id, [2239](#)
  - use\_facet, [2239](#)
- std::lock\_guard< \_Mutex >, [2357](#)
- std::logic\_error, [2358](#)
  - logic\_error, [2358](#)
  - what, [2359](#)
- std::logical\_and< \_Tp >, [2359](#)
  - first\_argument\_type, [2359](#)
  - result\_type, [2359](#)
  - second\_argument\_type, [2360](#)
- std::logical\_and< void >, [2360](#)
  - first\_argument\_type, [2361](#)
  - result\_type, [2361](#)
  - second\_argument\_type, [2361](#)
- std::logical\_not< \_Tp >, [2361](#)
  - argument\_type, [2362](#)
  - result\_type, [2362](#)
- std::logical\_not< void >, [2362](#)
  - argument\_type, [2363](#)
  - result\_type, [2363](#)
- std::logical\_or< \_Tp >, [2363](#)
  - first\_argument\_type, [2364](#)
  - result\_type, [2364](#)



- second\_argument\_type, 2364
- std::logical\_or< void >, 2364
  - first\_argument\_type, 2365
  - result\_type, 2365
  - second\_argument\_type, 2365
- std::lognormal\_distribution< \_RealType >, 2365
  - max, 2366
  - min, 2366
  - operator<<, 2367
  - operator>>, 2368
  - operator(), 2366
  - operator==, 2367
  - param, 2367
  - reset, 2367
  - result\_type, 2366
- std::lognormal\_distribution< \_RealType >::param\_type, 2724
- std::make\_signed< \_Tp >, 2375
- std::make\_unsigned< \_Tp >, 2375
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2379
  - \_\_attribute, 2385
  - ~map, 2385
  - at, 2386
  - begin, 2386
  - cbegin, 2386
  - cend, 2386
  - clear, 2387
  - contains, 2387
  - count, 2387, 2388
  - crbegin, 2388
  - crend, 2388
  - emplace, 2388
  - emplace\_hint, 2389
  - empty, 2389
  - end, 2390
  - equal\_range, 2390, 2391
  - erase, 2392
  - find, 2393, 2394
  - get\_allocator, 2394
  - insert, 2394–2397
  - insert\_or\_assign, 2398
  - key\_comp, 2399
  - lower\_bound, 2399, 2400
  - map, 2382–2385
  - max\_size, 2400
  - operator=, 2400, 2401
  - operator[], 2401
  - rbegin, 2401, 2402
  - rend, 2402
  - size, 2402
  - swap, 2402
  - upper\_bound, 2402, 2403
  - value\_comp, 2404
- std::mask\_array< \_Tp >, 2404
- operator<=, 2406
- operator>=, 2406
- operator+=, 2406
- operator-=, 2406
- operator/=: 2406
- operator%=: 2405
- operator&=: 2405
- operator\*=, 2406
- operator^=: 2406
- operator|=, 2406
- std::match\_results< \_Bi\_iter, \_Alloc >, 2408
  - ~match\_results, 2411
  - begin, 2411
  - cbegin, 2411
  - cend, 2412
  - empty, 2412
  - end, 2412
  - format, 2412, 2413
  - get\_allocator, 2413
  - length, 2413
  - match\_results, 2410, 2411
  - max\_size, 2413
  - operator=, 2414
  - operator[], 2414
  - position, 2414
  - prefix, 2415
  - ready, 2415
  - size, 2415
  - str, 2415
  - suffix, 2416
  - swap, 2416
- std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 2418
  - first\_argument\_type, 2419
  - result\_type, 2419
  - second\_argument\_type, 2419
- std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 2420
  - first\_argument\_type, 2420
  - result\_type, 2420
  - second\_argument\_type, 2420
- std::mem\_fun\_ref\_t< \_Ret, \_Tp >, 2421
  - argument\_type, 2421
  - result\_type, 2421
- std::mem\_fun\_t< \_Ret, \_Tp >, 2422
  - argument\_type, 2422
  - result\_type, 2422
- std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 2423
  - discard, 2426
  - max, 2426
  - mersenne\_twister\_engine, 2426
  - min, 2426
  - operator<<, 2427
  - operator>>, 2427



- operator==, 2427
- result\_type, 2425
- std::messages< \_CharT >, 2428
  - ~messages, 2430
  - char\_type, 2430
  - do\_get, 2431
  - id, 2431
  - messages, 2430
  - string\_type, 2430
- std::messages\_base, 2431
- std::messages\_byname< \_CharT >, 2432
  - do\_get, 2433
  - id, 2433
- std::minus< \_Tp >, 2434
  - first\_argument\_type, 2434
  - result\_type, 2434
  - second\_argument\_type, 2434
- std::minus< void >, 2435
  - first\_argument\_type, 2435
  - result\_type, 2435
  - second\_argument\_type, 2435
- std::modulus< \_Tp >, 2436
  - first\_argument\_type, 2437
  - result\_type, 2437
  - second\_argument\_type, 2437
- std::modulus< void >, 2437
  - first\_argument\_type, 2438
  - result\_type, 2438
  - second\_argument\_type, 2438
- std::money\_base, 2438
- std::money\_get< \_CharT, \_InIter >, 2440
  - ~money\_get, 2441
  - char\_type, 2441
  - do\_get, 2442
  - get, 2442, 2443
  - id, 2444
  - iter\_type, 2441
  - money\_get, 2441
  - string\_type, 2441
- std::money\_put< \_CharT, \_OutIter >, 2444
  - ~money\_put, 2446
  - char\_type, 2445
  - do\_put, 2446
  - id, 2448
  - iter\_type, 2445
  - money\_put, 2446
  - put, 2447, 2448
  - string\_type, 2445
- std::moneypunct< \_CharT, \_Intl >, 2448
  - ~moneypunct, 2451
  - char\_type, 2450
  - curr\_symbol, 2452
  - decimal\_point, 2452
  - do\_curr\_symbol, 2452
  - do\_decimal\_point, 2452
  - do\_frac\_digits, 2453
  - do\_grouping, 2453
  - do\_neg\_format, 2453
  - do\_negative\_sign, 2453
  - do\_pos\_format, 2454
  - do\_positive\_sign, 2454
  - do\_thousands\_sep, 2454
  - frac\_digits, 2455
  - grouping, 2455
  - id, 2457
  - intl, 2457
  - moneypunct, 2451
  - neg\_format, 2455
  - negative\_sign, 2456
  - pos\_format, 2456
  - positive\_sign, 2456
  - string\_type, 2450
  - thousands\_sep, 2457
- std::moneypunct\_byname< \_CharT, \_Intl >, 2457
  - curr\_symbol, 2459
  - decimal\_point, 2459
  - do\_curr\_symbol, 2460
  - do\_decimal\_point, 2460
  - do\_frac\_digits, 2460
  - do\_grouping, 2461
  - do\_neg\_format, 2461
  - do\_negative\_sign, 2461
  - do\_pos\_format, 2461
  - do\_positive\_sign, 2462
  - do\_thousands\_sep, 2462
  - frac\_digits, 2462
  - grouping, 2463
  - id, 2465
  - neg\_format, 2463
  - negative\_sign, 2463
  - pos\_format, 2464
  - positive\_sign, 2464
  - thousands\_sep, 2464
- std::move\_iterator< \_Iterator >, 2466
- std::move\_only\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV  
noexcept(\_Noex)>, 2468
  - move\_only\_function, 2469, 2470
  - operator bool, 2470
  - operator(), 2470
  - operator=, 2470, 2471
  - operator==, 2471
  - swap, 2471
- std::move\_sentinel< \_Sent >, 2472
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2475
  - ~multimap, 2481
  - begin, 2481
  - cbegin, 2481
  - cend, 2481

- clear, 2481
- contains, 2482
- count, 2482, 2483
- cbegin, 2483
- crend, 2483
- emplace, 2483
- emplace\_hint, 2484
- empty, 2484
- end, 2484
- equal\_range, 2485, 2486
- erase, 2486, 2487
- find, 2488, 2489
- get\_allocator, 2489
- insert, 2489–2492
- key\_comp, 2493
- lower\_bound, 2493, 2494
- max\_size, 2494
- multimap, 2478–2480
- operator=, 2495
- rbegin, 2495
- rend, 2496
- size, 2496
- swap, 2496
- upper\_bound, 2496, 2497
- value\_comp, 2498
- std::multiplies< \_Tp >, 2498
  - first\_argument\_type, 2499
  - result\_type, 2499
  - second\_argument\_type, 2499
- std::multiplies< void >, 2499
  - first\_argument\_type, 2500
  - result\_type, 2500
  - second\_argument\_type, 2500
- std::multiset< \_Key, \_Compare, \_Alloc >, 2503
  - ~multiset, 2508
  - begin, 2509
  - cbegin, 2509
  - cend, 2509
  - clear, 2509
  - contains, 2509, 2510
  - count, 2510
  - cbegin, 2510
  - crend, 2511
  - emplace, 2511
  - emplace\_hint, 2511
  - empty, 2512
  - end, 2512
  - equal\_range, 2512, 2513
  - erase, 2514
  - find, 2515, 2516
  - get\_allocator, 2516
  - insert, 2516–2518
  - key\_comp, 2518
  - lower\_bound, 2518, 2519
  - max\_size, 2520
  - multiset, 2506–2508
  - operator=, 2520
  - rbegin, 2520
  - rend, 2521
  - size, 2521
  - swap, 2521
  - upper\_bound, 2521, 2522
  - value\_comp, 2522
- std::mutex, 2526
- std::negate< \_Tp >, 2526
  - argument\_type, 2527
  - result\_type, 2527
- std::negate< void >, 2527
  - argument\_type, 2528
  - result\_type, 2528
- std::negative\_binomial\_distribution< \_IntType >, 2528
  - k, 2530
  - max, 2530
  - min, 2530
  - operator<=, 2531
  - operator>=, 2531
  - operator(), 2530
  - operator==, 2531
  - p, 2530
  - param, 2530
  - reset, 2531
  - result\_type, 2530
- std::negative\_binomial\_distribution< \_IntType >::param\_type, 2725
- std::nested\_exception, 2532
  - nested\_exception, 2532
  - nested\_ptr, 2532
  - rethrow\_nested, 2532
- std::normal\_distribution< \_RealType >, 2535
  - max, 2536
  - mean, 2536
  - min, 2537
  - normal\_distribution, 2536
  - operator<=, 2538
  - operator>=, 2538
  - operator(), 2537
  - operator==, 2538
  - param, 2537
  - reset, 2537
  - result\_type, 2536
  - stddev, 2537
- std::normal\_distribution< \_RealType >::param\_type, 2725
- std::not\_equal\_to< \_Tp >, 2539
  - first\_argument\_type, 2539
  - result\_type, 2539
  - second\_argument\_type, 2540
- std::not\_equal\_to< void >, 2540

- first\_argument\_type, 2541
- result\_type, 2541
- second\_argument\_type, 2541
- std::num\_get< \_CharT, \_InIter >, 2544
  - ~num\_get, 2546
  - char\_type, 2546
  - do\_get, 2546–2552
  - get, 2553–2559
  - id, 2559
  - iter\_type, 2546
  - num\_get, 2546
- std::num\_put< \_CharT, \_OutIter >, 2559
  - ~num\_put, 2562
  - char\_type, 2561
  - do\_put, 2562–2566
  - id, 2572
  - iter\_type, 2561
  - num\_put, 2562
  - put, 2566–2571
- std::numeric\_limits< \_Tp >, 2572
  - denorm\_min, 2573
  - digits, 2574
  - digits10, 2575
  - epsilon, 2573
  - has\_denorm, 2575
  - has\_denorm\_loss, 2575
  - has\_infinity, 2575
  - has\_quiet\_NaN, 2575
  - has\_signaling\_NaN, 2575
  - infinity, 2573
  - is\_bounded, 2575
  - is\_exact, 2575
  - is\_iec559, 2575
  - is\_integer, 2575
  - is\_modulo, 2576
  - is\_signed, 2576
  - is\_specialized, 2576
  - lowest, 2573
  - max, 2574
  - max\_digits10, 2576
  - max\_exponent, 2576
  - max\_exponent10, 2576
  - min, 2574
  - min\_exponent, 2576
  - min\_exponent10, 2576
  - quiet\_NaN, 2574
  - radix, 2576
  - round\_error, 2574
  - round\_style, 2576
  - signaling\_NaN, 2574
  - tinyness\_before, 2577
  - traps, 2577
- std::numeric\_limits< bool >, 2577
  - denorm\_min, 2579
- digits, 2580
- digits10, 2580
- epsilon, 2579
- has\_denorm, 2580
- has\_denorm\_loss, 2580
- has\_infinity, 2580
- has\_quiet\_NaN, 2580
- has\_signaling\_NaN, 2580
- infinity, 2579
- is\_bounded, 2580
- is\_exact, 2580
- is\_iec559, 2581
- is\_integer, 2581
- is\_modulo, 2581
- is\_signed, 2581
- is\_specialized, 2581
- lowest, 2579
- max, 2579
- max\_digits10, 2581
- max\_exponent, 2581
- max\_exponent10, 2581
- min, 2579
- min\_exponent, 2581
- min\_exponent10, 2582
- quiet\_NaN, 2579
- radix, 2582
- round\_error, 2579
- round\_style, 2582
- signaling\_NaN, 2580
- tinyness\_before, 2582
- traps, 2582
- std::numeric\_limits< char >, 2582
  - denorm\_min, 2584
  - digits, 2585
  - digits10, 2585
  - epsilon, 2584
  - has\_denorm, 2585
  - has\_denorm\_loss, 2585
  - has\_infinity, 2585
  - has\_quiet\_NaN, 2585
  - has\_signaling\_NaN, 2585
  - infinity, 2584
  - is\_bounded, 2586
  - is\_exact, 2586
  - is\_iec559, 2586
  - is\_integer, 2586
  - is\_modulo, 2586
  - is\_signed, 2586
  - is\_specialized, 2586
  - lowest, 2584
  - max, 2584
  - max\_digits10, 2586
  - max\_exponent, 2586
  - max\_exponent10, 2586

- min, [2584](#)
- min\_exponent, [2587](#)
- min\_exponent10, [2587](#)
- quiet\_NaN, [2585](#)
- radix, [2587](#)
- round\_error, [2585](#)
- round\_style, [2587](#)
- signaling\_NaN, [2585](#)
- tinyness\_before, [2587](#)
- traps, [2587](#)
- std::numeric\_limits< char16\_t >, [2587](#)
  - denorm\_min, [2589](#)
  - digits, [2590](#)
  - digits10, [2590](#)
  - epsilon, [2589](#)
  - has\_denorm, [2590](#)
  - has\_denorm\_loss, [2591](#)
  - has\_infinity, [2591](#)
  - has\_quiet\_NaN, [2591](#)
  - has\_signaling\_NaN, [2591](#)
  - infinity, [2589](#)
  - is\_bounded, [2591](#)
  - is\_exact, [2591](#)
  - is\_iec559, [2591](#)
  - is\_integer, [2591](#)
  - is\_modulo, [2591](#)
  - is\_signed, [2591](#)
  - is\_specialized, [2592](#)
  - lowest, [2590](#)
  - max, [2590](#)
  - max\_digits10, [2592](#)
  - max\_exponent, [2592](#)
  - max\_exponent10, [2592](#)
  - min, [2590](#)
  - min\_exponent, [2592](#)
  - min\_exponent10, [2592](#)
  - quiet\_NaN, [2590](#)
  - radix, [2592](#)
  - round\_error, [2590](#)
  - round\_style, [2592](#)
  - signaling\_NaN, [2590](#)
  - tinyness\_before, [2592](#)
  - traps, [2592](#)
- std::numeric\_limits< char32\_t >, [2593](#)
  - denorm\_min, [2595](#)
  - digits, [2596](#)
  - digits10, [2596](#)
  - epsilon, [2595](#)
  - has\_denorm, [2596](#)
  - has\_denorm\_loss, [2596](#)
  - has\_infinity, [2596](#)
  - has\_quiet\_NaN, [2596](#)
  - has\_signaling\_NaN, [2596](#)
  - infinity, [2595](#)
  - is\_bounded, [2596](#)
  - is\_exact, [2596](#)
  - is\_iec559, [2596](#)
  - is\_integer, [2596](#)
  - is\_modulo, [2597](#)
  - is\_signed, [2597](#)
  - is\_specialized, [2597](#)
  - lowest, [2595](#)
  - max, [2595](#)
  - max\_digits10, [2597](#)
  - max\_exponent, [2597](#)
  - max\_exponent10, [2597](#)
  - min, [2595](#)
  - min\_exponent, [2597](#)
  - min\_exponent10, [2597](#)
  - quiet\_NaN, [2595](#)
  - radix, [2597](#)
  - round\_error, [2595](#)
  - round\_style, [2597](#)
  - signaling\_NaN, [2595](#)
  - tinyness\_before, [2598](#)
  - traps, [2598](#)
- std::numeric\_limits< double >, [2598](#)
  - denorm\_min, [2600](#)
  - digits, [2601](#)
  - digits10, [2601](#)
  - epsilon, [2600](#)
  - has\_denorm, [2601](#)
  - has\_denorm\_loss, [2601](#)
  - has\_infinity, [2601](#)
  - has\_quiet\_NaN, [2601](#)
  - has\_signaling\_NaN, [2601](#)
  - infinity, [2600](#)
  - is\_bounded, [2601](#)
  - is\_exact, [2601](#)
  - is\_iec559, [2602](#)
  - is\_integer, [2602](#)
  - is\_modulo, [2602](#)
  - is\_signed, [2602](#)
  - is\_specialized, [2602](#)
  - lowest, [2600](#)
  - max, [2600](#)
  - max\_digits10, [2602](#)
  - max\_exponent, [2602](#)
  - max\_exponent10, [2602](#)
  - min, [2600](#)
  - min\_exponent, [2602](#)
  - min\_exponent10, [2603](#)
  - quiet\_NaN, [2600](#)
  - radix, [2603](#)
  - round\_error, [2600](#)
  - round\_style, [2603](#)
  - signaling\_NaN, [2601](#)
  - tinyness\_before, [2603](#)

- traps, [2603](#)
- std::numeric\_limits< float >, [2603](#)
  - denorm\_min, [2605](#)
  - digits, [2606](#)
  - digits10, [2606](#)
  - epsilon, [2605](#)
  - has\_denorm, [2606](#)
  - has\_denorm\_loss, [2606](#)
  - has\_infinity, [2606](#)
  - has\_quiet\_NaN, [2606](#)
  - has\_signaling\_NaN, [2606](#)
  - infinity, [2605](#)
  - is\_bounded, [2607](#)
  - is\_exact, [2607](#)
  - is\_iec559, [2607](#)
  - is\_integer, [2607](#)
  - is\_modulo, [2607](#)
  - is\_signed, [2607](#)
  - is\_specialized, [2607](#)
  - lowest, [2605](#)
  - max, [2605](#)
  - max\_digits10, [2607](#)
  - max\_exponent, [2607](#)
  - max\_exponent10, [2607](#)
  - min, [2605](#)
  - min\_exponent, [2608](#)
  - min\_exponent10, [2608](#)
  - quiet\_NaN, [2606](#)
  - radix, [2608](#)
  - round\_error, [2606](#)
  - round\_style, [2608](#)
  - signaling\_NaN, [2606](#)
  - tinyness\_before, [2608](#)
  - traps, [2608](#)
- std::numeric\_limits< int >, [2608](#)
  - denorm\_min, [2610](#)
  - digits, [2611](#)
  - digits10, [2611](#)
  - epsilon, [2610](#)
  - has\_denorm, [2611](#)
  - has\_denorm\_loss, [2611](#)
  - has\_infinity, [2612](#)
  - has\_quiet\_NaN, [2612](#)
  - has\_signaling\_NaN, [2612](#)
  - infinity, [2610](#)
  - is\_bounded, [2612](#)
  - is\_exact, [2612](#)
  - is\_iec559, [2612](#)
  - is\_integer, [2612](#)
  - is\_modulo, [2612](#)
  - is\_signed, [2612](#)
  - is\_specialized, [2612](#)
  - lowest, [2611](#)
  - max, [2611](#)
  - max\_digits10, [2613](#)
  - max\_exponent, [2613](#)
  - max\_exponent10, [2613](#)
  - min, [2611](#)
  - min\_exponent, [2613](#)
  - min\_exponent10, [2613](#)
  - quiet\_NaN, [2611](#)
  - radix, [2613](#)
  - round\_error, [2611](#)
  - round\_style, [2613](#)
  - signaling\_NaN, [2611](#)
  - tinyness\_before, [2613](#)
  - traps, [2613](#)
- std::numeric\_limits< long >, [2614](#)
  - denorm\_min, [2615](#)
  - digits, [2616](#)
  - digits10, [2616](#)
  - epsilon, [2615](#)
  - has\_denorm, [2617](#)
  - has\_denorm\_loss, [2617](#)
  - has\_infinity, [2617](#)
  - has\_quiet\_NaN, [2617](#)
  - has\_signaling\_NaN, [2617](#)
  - infinity, [2616](#)
  - is\_bounded, [2617](#)
  - is\_exact, [2617](#)
  - is\_iec559, [2617](#)
  - is\_integer, [2617](#)
  - is\_modulo, [2617](#)
  - is\_signed, [2618](#)
  - is\_specialized, [2618](#)
  - lowest, [2616](#)
  - max, [2616](#)
  - max\_digits10, [2618](#)
  - max\_exponent, [2618](#)
  - max\_exponent10, [2618](#)
  - min, [2616](#)
  - min\_exponent, [2618](#)
  - min\_exponent10, [2618](#)
  - quiet\_NaN, [2616](#)
  - radix, [2618](#)
  - round\_error, [2616](#)
  - round\_style, [2618](#)
  - signaling\_NaN, [2616](#)
  - tinyness\_before, [2619](#)
  - traps, [2619](#)
- std::numeric\_limits< long double >, [2619](#)
  - denorm\_min, [2621](#)
  - digits, [2622](#)
  - digits10, [2622](#)
  - epsilon, [2621](#)
  - has\_denorm, [2622](#)
  - has\_denorm\_loss, [2622](#)
  - has\_infinity, [2622](#)

- has\_quiet\_NaN, [2622](#)
- has\_signaling\_NaN, [2622](#)
- infinity, [2621](#)
- is\_bounded, [2622](#)
- is\_exact, [2622](#)
- is\_iec559, [2623](#)
- is\_integer, [2623](#)
- is\_modulo, [2623](#)
- is\_signed, [2623](#)
- is\_specialized, [2623](#)
- lowest, [2621](#)
- max, [2621](#)
- max\_digits10, [2623](#)
- max\_exponent, [2623](#)
- max\_exponent10, [2623](#)
- min, [2621](#)
- min\_exponent, [2623](#)
- min\_exponent10, [2624](#)
- quiet\_NaN, [2621](#)
- radix, [2624](#)
- round\_error, [2621](#)
- round\_style, [2624](#)
- signaling\_NaN, [2622](#)
- tinyness\_before, [2624](#)
- traps, [2624](#)
- std::numeric\_limits< long long >, [2624](#)
  - denorm\_min, [2626](#)
  - digits, [2627](#)
  - digits10, [2627](#)
  - epsilon, [2626](#)
  - has\_denorm, [2627](#)
  - has\_denorm\_loss, [2627](#)
  - has\_infinity, [2627](#)
  - has\_quiet\_NaN, [2627](#)
  - has\_signaling\_NaN, [2627](#)
  - infinity, [2626](#)
  - is\_bounded, [2628](#)
  - is\_exact, [2628](#)
  - is\_iec559, [2628](#)
  - is\_integer, [2628](#)
  - is\_modulo, [2628](#)
  - is\_signed, [2628](#)
  - is\_specialized, [2628](#)
  - lowest, [2626](#)
  - max, [2626](#)
  - max\_digits10, [2628](#)
  - max\_exponent, [2628](#)
  - max\_exponent10, [2628](#)
  - min, [2626](#)
  - min\_exponent, [2629](#)
  - min\_exponent10, [2629](#)
  - quiet\_NaN, [2627](#)
  - radix, [2629](#)
  - round\_error, [2627](#)
  - round\_style, [2629](#)
  - signaling\_NaN, [2627](#)
  - tinyness\_before, [2629](#)
  - traps, [2629](#)
- std::numeric\_limits< short >, [2629](#)
  - denorm\_min, [2631](#)
  - digits, [2632](#)
  - digits10, [2632](#)
  - epsilon, [2631](#)
  - has\_denorm, [2632](#)
  - has\_denorm\_loss, [2633](#)
  - has\_infinity, [2633](#)
  - has\_quiet\_NaN, [2633](#)
  - has\_signaling\_NaN, [2633](#)
  - infinity, [2631](#)
  - is\_bounded, [2633](#)
  - is\_exact, [2633](#)
  - is\_iec559, [2633](#)
  - is\_integer, [2633](#)
  - is\_modulo, [2633](#)
  - is\_signed, [2633](#)
  - is\_specialized, [2634](#)
  - lowest, [2632](#)
  - max, [2632](#)
  - max\_digits10, [2634](#)
  - max\_exponent, [2634](#)
  - max\_exponent10, [2634](#)
  - min, [2632](#)
  - min\_exponent, [2634](#)
  - min\_exponent10, [2634](#)
  - quiet\_NaN, [2632](#)
  - radix, [2634](#)
  - round\_error, [2632](#)
  - round\_style, [2634](#)
  - signaling\_NaN, [2632](#)
  - tinyness\_before, [2634](#)
  - traps, [2634](#)
- std::numeric\_limits< signed char >, [2635](#)
  - denorm\_min, [2637](#)
  - digits, [2638](#)
  - digits10, [2638](#)
  - epsilon, [2637](#)
  - has\_denorm, [2638](#)
  - has\_denorm\_loss, [2638](#)
  - has\_infinity, [2638](#)
  - has\_quiet\_NaN, [2638](#)
  - has\_signaling\_NaN, [2638](#)
  - infinity, [2637](#)
  - is\_bounded, [2638](#)
  - is\_exact, [2638](#)
  - is\_iec559, [2638](#)
  - is\_integer, [2638](#)
  - is\_modulo, [2639](#)
  - is\_signed, [2639](#)

- is\_specialized, [2639](#)
- lowest, [2637](#)
- max, [2637](#)
- max\_digits10, [2639](#)
- max\_exponent, [2639](#)
- max\_exponent10, [2639](#)
- min, [2637](#)
- min\_exponent, [2639](#)
- min\_exponent10, [2639](#)
- quiet\_NaN, [2637](#)
- radix, [2639](#)
- round\_error, [2637](#)
- round\_style, [2639](#)
- signaling\_NaN, [2637](#)
- tinyness\_before, [2640](#)
- traps, [2640](#)
- std::numeric\_limits< unsigned char >, [2640](#)
  - denorm\_min, [2642](#)
  - digits, [2643](#)
  - digits10, [2643](#)
  - epsilon, [2642](#)
  - has\_denorm, [2643](#)
  - has\_denorm\_loss, [2643](#)
  - has\_infinity, [2643](#)
  - has\_quiet\_NaN, [2643](#)
  - has\_signaling\_NaN, [2643](#)
  - infinity, [2642](#)
  - is\_bounded, [2643](#)
  - is\_exact, [2643](#)
  - is\_iec559, [2644](#)
  - is\_integer, [2644](#)
  - is\_modulo, [2644](#)
  - is\_signed, [2644](#)
  - is\_specialized, [2644](#)
  - lowest, [2642](#)
  - max, [2642](#)
  - max\_digits10, [2644](#)
  - max\_exponent, [2644](#)
  - max\_exponent10, [2644](#)
  - min, [2642](#)
  - min\_exponent, [2644](#)
  - min\_exponent10, [2645](#)
  - quiet\_NaN, [2642](#)
  - radix, [2645](#)
  - round\_error, [2642](#)
  - round\_style, [2645](#)
  - signaling\_NaN, [2643](#)
  - tinyness\_before, [2645](#)
  - traps, [2645](#)
- std::numeric\_limits< unsigned int >, [2645](#)
  - denorm\_min, [2647](#)
  - digits, [2648](#)
  - digits10, [2648](#)
  - epsilon, [2647](#)
- has\_denorm, [2648](#)
- has\_denorm\_loss, [2648](#)
- has\_infinity, [2648](#)
- has\_quiet\_NaN, [2648](#)
- has\_signaling\_NaN, [2648](#)
- infinity, [2647](#)
- is\_bounded, [2649](#)
- is\_exact, [2649](#)
- is\_iec559, [2649](#)
- is\_integer, [2649](#)
- is\_modulo, [2649](#)
- is\_signed, [2649](#)
- is\_specialized, [2649](#)
- lowest, [2647](#)
- max, [2647](#)
- max\_digits10, [2649](#)
- max\_exponent, [2649](#)
- max\_exponent10, [2649](#)
- min, [2647](#)
- min\_exponent, [2650](#)
- min\_exponent10, [2650](#)
- quiet\_NaN, [2648](#)
- radix, [2650](#)
- round\_error, [2648](#)
- round\_style, [2650](#)
- signaling\_NaN, [2648](#)
- tinyness\_before, [2650](#)
- traps, [2650](#)
- std::numeric\_limits< unsigned long >, [2650](#)
  - denorm\_min, [2652](#)
  - digits, [2653](#)
  - digits10, [2653](#)
  - epsilon, [2652](#)
  - has\_denorm, [2653](#)
  - has\_denorm\_loss, [2654](#)
  - has\_infinity, [2654](#)
  - has\_quiet\_NaN, [2654](#)
  - has\_signaling\_NaN, [2654](#)
  - infinity, [2652](#)
  - is\_bounded, [2654](#)
  - is\_exact, [2654](#)
  - is\_iec559, [2654](#)
  - is\_integer, [2654](#)
  - is\_modulo, [2654](#)
  - is\_signed, [2654](#)
  - is\_specialized, [2655](#)
  - lowest, [2653](#)
  - max, [2653](#)
  - max\_digits10, [2655](#)
  - max\_exponent, [2655](#)
  - max\_exponent10, [2655](#)
  - min, [2653](#)
  - min\_exponent, [2655](#)
  - min\_exponent10, [2655](#)

- quiet\_NaN, [2653](#)
- radix, [2655](#)
- round\_error, [2653](#)
- round\_style, [2655](#)
- signaling\_NaN, [2653](#)
- tinyness\_before, [2655](#)
- traps, [2655](#)
- std::numeric\_limits< unsigned long long >, [2656](#)
  - denorm\_min, [2658](#)
  - digits, [2659](#)
  - digits10, [2659](#)
  - epsilon, [2658](#)
  - has\_denorm, [2659](#)
  - has\_denorm\_loss, [2659](#)
  - has\_infinity, [2659](#)
  - has\_quiet\_NaN, [2659](#)
  - has\_signaling\_NaN, [2659](#)
  - infinity, [2658](#)
  - is\_bounded, [2659](#)
  - is\_exact, [2659](#)
  - is\_iec559, [2659](#)
  - is\_integer, [2659](#)
  - is\_modulo, [2660](#)
  - is\_signed, [2660](#)
  - is\_specialized, [2660](#)
  - lowest, [2658](#)
  - max, [2658](#)
  - max\_digits10, [2660](#)
  - max\_exponent, [2660](#)
  - max\_exponent10, [2660](#)
  - min, [2658](#)
  - min\_exponent, [2660](#)
  - min\_exponent10, [2660](#)
  - quiet\_NaN, [2658](#)
  - radix, [2660](#)
  - round\_error, [2658](#)
  - round\_style, [2660](#)
  - signaling\_NaN, [2658](#)
  - tinyness\_before, [2661](#)
  - traps, [2661](#)
- std::numeric\_limits< unsigned short >, [2661](#)
  - denorm\_min, [2663](#)
  - digits, [2664](#)
  - digits10, [2664](#)
  - epsilon, [2663](#)
  - has\_denorm, [2664](#)
  - has\_denorm\_loss, [2664](#)
  - has\_infinity, [2664](#)
  - has\_quiet\_NaN, [2664](#)
  - has\_signaling\_NaN, [2664](#)
  - infinity, [2663](#)
  - is\_bounded, [2664](#)
  - is\_exact, [2664](#)
  - is\_iec559, [2665](#)
- is\_integer, [2665](#)
- is\_modulo, [2665](#)
- is\_signed, [2665](#)
- is\_specialized, [2665](#)
- lowest, [2663](#)
- max, [2663](#)
- max\_digits10, [2665](#)
- max\_exponent, [2665](#)
- max\_exponent10, [2665](#)
- min, [2663](#)
- min\_exponent, [2665](#)
- min\_exponent10, [2666](#)
- quiet\_NaN, [2663](#)
- radix, [2666](#)
- round\_error, [2663](#)
- round\_style, [2666](#)
- signaling\_NaN, [2664](#)
- tinyness\_before, [2666](#)
- traps, [2666](#)
- std::numeric\_limits< wchar\_t >, [2666](#)
  - denorm\_min, [2668](#)
  - digits, [2669](#)
  - digits10, [2669](#)
  - epsilon, [2668](#)
  - has\_denorm, [2669](#)
  - has\_denorm\_loss, [2669](#)
  - has\_infinity, [2669](#)
  - has\_quiet\_NaN, [2669](#)
  - has\_signaling\_NaN, [2669](#)
  - infinity, [2668](#)
  - is\_bounded, [2670](#)
  - is\_exact, [2670](#)
  - is\_iec559, [2670](#)
  - is\_integer, [2670](#)
  - is\_modulo, [2670](#)
  - is\_signed, [2670](#)
  - is\_specialized, [2670](#)
  - lowest, [2668](#)
  - max, [2668](#)
  - max\_digits10, [2670](#)
  - max\_exponent, [2670](#)
  - max\_exponent10, [2670](#)
  - min, [2668](#)
  - min\_exponent, [2671](#)
  - min\_exponent10, [2671](#)
  - quiet\_NaN, [2669](#)
  - radix, [2671](#)
  - round\_error, [2669](#)
  - round\_style, [2671](#)
  - signaling\_NaN, [2669](#)
  - tinyness\_before, [2671](#)
  - traps, [2671](#)
- std::num\_punct< \_CharT >, [2671](#)
  - ~num\_punct, [2674](#)



- char\_type, 2673
- decimal\_point, 2674
- do\_decimal\_point, 2674
- do\_falsename, 2675
- do\_grouping, 2675
- do\_thousands\_sep, 2675
- do\_truename, 2675
- falsename, 2676
- grouping, 2676
- id, 2677
- numpunct, 2673, 2674
- string\_type, 2673
- thousands\_sep, 2676
- truename, 2676
- std::numpunct\_byname< \_CharT >, 2677
  - decimal\_point, 2678
  - do\_decimal\_point, 2679
  - do\_falsename, 2679
  - do\_grouping, 2679
  - do\_thousands\_sep, 2679
  - do\_truename, 2680
  - falsename, 2680
  - grouping, 2680
  - id, 2681
  - thousands\_sep, 2680
  - truename, 2681
- std::once\_flag, 2683
  - call\_once, 2684
  - once\_flag, 2684
  - operator=, 2684
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 2686
  - char\_type, 2687
  - difference\_type, 2687
  - iterator\_category, 2687
  - operator=, 2688
  - ostream\_iterator, 2688
  - ostream\_type, 2687
  - pointer, 2687
  - reference, 2687
  - traits\_type, 2687
  - value\_type, 2687
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 2689
  - char\_type, 2690
  - difference\_type, 2690
  - failed, 2691
  - iterator\_category, 2690
  - operator++, 2692
  - operator=, 2692
  - operator\*, 2692
  - ostream\_type, 2690
  - ostreambuf\_iterator, 2691
  - pointer, 2691
  - reference, 2691
  - streambuf\_type, 2691
- traits\_type, 2691
- value\_type, 2691
- std::out\_of\_range, 2692
  - what, 2693
- std::output\_iterator\_tag, 2693
- std::overflow\_error, 2701
  - what, 2702
- std::owner\_less< \_Tp >, 2702
- std::owner\_less< shared\_ptr< \_Tp > >, 2703
  - first\_argument\_type, 2704
  - result\_type, 2704
  - second\_argument\_type, 2704
- std::owner\_less< void >, 2704
  - first\_argument\_type, 2705
  - result\_type, 2705
  - second\_argument\_type, 2705
- std::owner\_less< weak\_ptr< \_Tp > >, 2706
  - first\_argument\_type, 2706
  - result\_type, 2706
  - second\_argument\_type, 2706
- std::packaged\_task< \_Res(\_ArgTypes...)>, 2706
- std::pair< \_T1, \_T2 >, 2710
  - first, 2714
  - first\_type, 2712
  - pair, 2713
  - second, 2714
  - second\_type, 2712
  - swap, 2714
- std::piecewise\_constant\_distribution< \_RealType >, 2741
  - densities, 2743
  - intervals, 2743
  - max, 2743
  - min, 2743
  - operator<=, 2745
  - operator>=, 2745
  - operator(), 2743
  - operator==, 2745
  - param, 2743
  - reset, 2745
  - result\_type, 2743
- std::piecewise\_constant\_distribution< \_RealType >::param\_type, 2726
- std::piecewise\_construct\_t, 2746
- std::piecewise\_linear\_distribution< \_RealType >, 2746
  - densities, 2747
  - intervals, 2747
  - max, 2747
  - min, 2747
  - operator<=, 2748
  - operator>=, 2749
  - operator(), 2748
  - operator==, 2749
  - param, 2748
  - reset, 2748

- result\_type, 2747
- std::piecewise\_linear\_distribution<\_RealType>::param\_type, 2727
- std::placeholders, 708
- std::plus<\_Tp>, 2749
  - first\_argument\_type, 2750
  - operator(), 2751
  - result\_type, 2750
  - second\_argument\_type, 2750
- std::pmr::memory\_resource, 2423
- std::pmr::monotonic\_buffer\_resource, 2465
  - do\_allocate, 2466
  - do\_deallocate, 2466
  - do\_is\_equal, 2466
- std::pmr::polymorphic\_allocator<\_Tp>, 2761
- std::pmr::pool\_options, 2762
  - max\_blocks\_per\_chunk, 2763
- std::pmr::unsynchronized\_pool\_resource, 3277
  - do\_allocate, 3278
  - do\_deallocate, 3278
  - do\_is\_equal, 3278
- std::pointer\_to\_binary\_function<\_Arg1, \_Arg2, \_Result>, 2751
  - first\_argument\_type, 2752
  - result\_type, 2752
  - second\_argument\_type, 2752
- std::pointer\_to\_unary\_function<\_Arg, \_Result>, 2753
  - argument\_type, 2754
  - result\_type, 2754
- std::pointer\_traits<\_Ptr>, 2754
  - difference\_type, 2754
  - element\_type, 2754
  - pointer, 2754
  - pointer\_to, 2755
  - rebind, 2755
- std::pointer\_traits<\_Tp\*>, 2755
  - difference\_type, 2756
  - element\_type, 2756
  - pointer, 2756
  - pointer\_to, 2757
  - rebind, 2756
- std::poisson\_distribution<\_IntType>, 2758
  - max, 2759
  - mean, 2759
  - min, 2759
  - operator<=, 2760
  - operator>=, 2761
  - operator(), 2759, 2760
  - operator==, 2761
  - param, 2760
  - reset, 2760
  - result\_type, 2759
- std::poisson\_distribution<\_IntType>::param\_type, 2727
- std::priority\_queue<\_Tp, \_Sequence, \_Compare>, 2764
  - empty, 2767
  - pop, 2767
  - priority\_queue, 2766
  - push, 2767
  - size, 2768
  - top, 2768
- std::promise<\_Res>, 2770
- std::promise<\_Res&>, 2771
- std::promise<void>, 2772
- std::queue<\_Tp, \_Sequence>, 2774
  - back, 2776
  - c, 2777
  - empty, 2776
  - front, 2776, 2777
  - pop, 2777
  - push, 2777
  - queue, 2776
  - size, 2777
- std::random\_access\_iterator\_tag, 2779
- std::random\_device, 2780
  - result\_type, 2781
- std::range\_error, 2781
  - what, 2782
- std::ranges::equal\_to, 2110
- std::ranges::greater, 2198
- std::ranges::greater\_equal, 2201
- std::ranges::less, 2314
- std::ranges::less\_equal, 2316
- std::ranges::not\_equal\_to, 2540
- std::ranges::subrange<\_It, \_Sent, \_Kind>, 3036
- std::ranges::view\_interface<\_Derived>, 3327
- std::rank<\_Tp>, 2788
- std::ratio<\_Num, \_Den>, 2789
- std::ratio\_equal<\_R1, \_R2>, 2789
- std::ratio\_greater<\_R1, \_R2>, 2790
- std::ratio\_greater\_equal<\_R1, \_R2>, 2791
- std::ratio\_less<\_R1, \_R2>, 2792
- std::ratio\_less\_equal<\_R1, \_R2>, 2793
- std::ratio\_not\_equal<\_R1, \_R2>, 2793
- std::raw\_storage\_iterator<\_OutputIterator, \_Tp>, 2794
  - difference\_type, 2795
  - iterator\_category, 2795
  - pointer, 2796
  - reference, 2796
  - value\_type, 2796
- std::recursive\_mutex, 2811
- std::recursive\_timed\_mutex, 2811
- std::reference\_wrapper<\_Tp>, 2813
  - cref, 2814
  - ref, 2814
- std::regex\_constants, 709
  - \_\_multiline, 715
  - \_\_polynomial, 715
  - awk, 715

- basic, 715
- collate, 715
- ECMAScript, 715
- egrep, 715
- error\_backref, 711
- error\_badbrace, 711
- error\_badrepeat, 711
- error\_brace, 711
- error\_brack, 711
- error\_collate, 711
- error\_complexity, 711
- error\_ctype, 711
- error\_escape, 712
- error\_paren, 712
- error\_range, 712
- error\_space, 712
- error\_stack, 712
- error\_type, 710
- extended, 716
- format\_default, 716
- format\_first\_only, 716
- format\_no\_copy, 716
- format\_sed, 716
- grep, 717
- icase, 717
- match\_any, 717
- match\_continuous, 717
- match\_default, 717
- match\_flag\_type, 710
- match\_not BOL, 717
- match\_not\_BOW, 717
- match\_not\_EOL, 717
- match\_not\_EOW, 718
- match\_not\_null, 718
- match\_prev\_avail, 718
- multiline, 718
- nosubs, 718
- operator&, 712
- operator&=, 712, 713
- operator~, 714
- operator^, 713
- operator^=, 713
- operator|, 714
- operator|=, 714
- optimize, 718
- syntax\_option\_type, 710
- std::regex\_error, 2814
  - code, 2815
  - regex\_error, 2815
  - what, 2816
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 2816
  - operator++, 2817, 2818
  - operator->, 2818
  - operator=, 2818
  - operator==, 2818
  - operator\*, 2817
  - regex\_iterator, 2817
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 2818
  - operator++, 2822
  - operator->, 2822
  - operator=, 2822
  - operator==, 2822
  - operator\*, 2822
  - regex\_token\_iterator, 2819–2821
- std::regex\_traits< \_Ch\_type >, 2823
  - getloc, 2824
  - imbue, 2824
  - isctype, 2824
  - length, 2825
  - lookup\_classname, 2825
  - lookup\_collatename, 2826
  - regex\_traits, 2824
  - transform, 2826
  - transform\_primary, 2827
  - translate, 2827
  - translate\_nocase, 2827
  - value, 2828
- std::rel\_ops, 718
  - operator!=, 719
  - operator<=, 719
  - operator>, 719
  - operator>=, 720
- std::remove\_all\_extents< \_Tp >, 2828
- std::remove\_const< \_Tp >, 2829
- std::remove\_cv< \_Tp >, 2829
- std::remove\_extent< \_Tp >, 2829
- std::remove\_pointer< \_Tp >, 2830
- std::remove\_reference< \_Tp >, 2830
- std::remove\_volatile< \_Tp >, 2830
- std::result\_of< \_Signature >, 2832
- std::reverse\_iterator< \_Iterator >, 2833
  - base, 2835
  - operator+, 2835
  - operator++, 2835
  - operator+=, 2836
  - operator-, 2836
  - operator->, 2837
  - operator--, 2836
  - operator=, 2836
  - operator[], 2837
  - operator\*, 2835
  - reverse\_iterator, 2834
- std::runtime\_error, 2842
  - runtime\_error, 2843
  - what, 2843

std::scoped\_allocator\_adaptor< \_OuterAlloc, \_InnerAl-  
     locs >, 2858  
 std::seed\_seq, 2859  
     result\_type, 2860  
     seed\_seq, 2860  
 std::set< \_Key, \_Compare, \_Alloc >, 2928  
     ~set, 2936  
     allocator\_type, 2931  
     begin, 2936  
     cbegin, 2936  
     cend, 2936  
     clear, 2936  
     const\_iterator, 2931  
     const\_pointer, 2931  
     const\_reference, 2931  
     const\_reverse\_iterator, 2931  
     contains, 2936, 2937  
     count, 2937  
     crbegin, 2938  
     crend, 2938  
     difference\_type, 2932  
     emplace, 2938  
     emplace\_hint, 2938  
     empty, 2939  
     end, 2939  
     equal\_range, 2939, 2940  
     erase, 2941, 2942  
     find, 2942, 2943  
     get\_allocator, 2943  
     insert, 2944, 2945  
     iterator, 2932  
     key\_comp, 2945  
     key\_compare, 2932  
     key\_type, 2932  
     lower\_bound, 2945, 2946  
     max\_size, 2948  
     operator=, 2948  
     pointer, 2932  
     rbegin, 2948  
     reference, 2932  
     rend, 2949  
     reverse\_iterator, 2932  
     set, 2933–2935  
     size, 2949  
     size\_type, 2932  
     swap, 2949  
     upper\_bound, 2949, 2950  
     value\_comp, 2951  
     value\_compare, 2933  
     value\_type, 2933  
 std::shared\_future< \_Res >, 2951  
     \_M\_get\_result, 2952  
     get, 2952  
     shared\_future, 2952  
 std::shared\_future< \_Res & >, 2953  
     \_M\_get\_result, 2954  
     get, 2954, 2955  
     shared\_future, 2954  
 std::shared\_future< void >, 2955  
     \_M\_get\_result, 2957  
     get, 2957  
     shared\_future, 2956  
 std::shared\_lock< \_Mutex >, 2957  
 std::shared\_ptr< \_Tp >, 2958  
     element\_type, 2960  
     get, 2966  
     operator bool, 2966  
     owner\_before, 2966, 2967  
     shared\_ptr, 2960–2966  
     swap, 2967  
     unique, 2967  
     use\_count, 2967  
 std::shared\_timed\_mutex, 2967  
 std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k  
     >, 2968  
     base, 2970  
     discard, 2970  
     max, 2970  
     min, 2971  
     operator<=, 2971  
     operator>, 2972  
     operator(), 2971  
     operator==, 2972  
     result\_type, 2969  
     seed, 2971  
     shuffle\_order\_engine, 2969, 2970  
 std::slice, 2973  
     operator==, 2973  
 std::slice\_array< \_Tp >, 2973  
     operator<=, 2975  
     operator>=, 2975  
     operator+=, 2975  
     operator-=, 2975  
     operator/=, 2975  
     operator%=", 2975  
     operator&=", 2975  
     operator\*=", 2975  
     operator^=", 2976  
     operator|=, 2976  
 std::stack< \_Tp, \_Sequence >, 2984  
     empty, 2986  
     pop, 2986  
     push, 2986  
     size, 2986  
     stack, 2986  
     top, 2986, 2987  
 std::student\_t\_distribution< \_RealType >, 3030  
     max, 3031

- min, 3031
- operator<<, 3032
- operator>>, 3033
- operator(), 3031
- operator==, 3032
- param, 3032
- reset, 3032
- result\_type, 3031
- std::student\_t\_distribution< \_RealType >::param\_type, 2728
- std::sub\_match< \_Bilter >, 3033
  - compare, 3035
  - length, 3036
  - operator string\_type, 3036
  - str, 3036
  - swap, 3036
- std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3038
  - discard, 3040
  - max, 3040
  - min, 3040
  - operator<<, 3041
  - operator>>, 3041
  - operator(), 3040
  - operator==, 3041
  - result\_type, 3039
  - seed, 3040
  - subtract\_with\_carry\_engine, 3039
- std::system\_error, 3045
  - what, 3045
- std::this\_thread, 720
  - get\_id, 721
  - sleep\_for, 721
  - sleep\_until, 721
  - yield, 721
- std::thread, 3051
  - native\_handle, 3052
- std::thread::id, 2240
- std::time\_base, 3060
- std::time\_get< \_CharT, \_InIter >, 3061
  - ~time\_get, 3063
  - char\_type, 3063
  - date\_order, 3063
  - do\_date\_order, 3063
  - do\_get, 3064
  - do\_get\_date, 3064
  - do\_get\_monthname, 3065
  - do\_get\_time, 3066
  - do\_get\_weekday, 3066
  - do\_get\_year, 3067
  - get, 3067, 3068
  - get\_date, 3069
  - get\_monthname, 3069
  - get\_time, 3070
  - get\_weekday, 3070
  - get\_year, 3071
  - id, 3072
  - iter\_type, 3063
  - time\_get, 3063
- std::time\_get\_byname< \_CharT, \_InIter >, 3072
  - date\_order, 3074
  - do\_date\_order, 3074
  - do\_get, 3074
  - do\_get\_date, 3075
  - do\_get\_monthname, 3075
  - do\_get\_time, 3076
  - do\_get\_weekday, 3077
  - do\_get\_year, 3077
  - get, 3078
  - get\_date, 3079
  - get\_monthname, 3080
  - get\_time, 3080
  - get\_weekday, 3081
  - get\_year, 3081
  - id, 3082
- std::time\_put< \_CharT, \_OutIter >, 3083
  - ~time\_put, 3085
  - char\_type, 3085
  - do\_put, 3085
  - id, 3087
  - iter\_type, 3085
  - put, 3086, 3087
  - time\_put, 3085
- std::time\_put\_byname< \_CharT, \_OutIter >, 3087
  - do\_put, 3089
  - id, 3090
  - put, 3089, 3090
- std::timed\_mutex, 3091
- std::to\_chars\_result, 3091
- std::tr1, 721
  - arg, 724
  - conf\_hyperg, 724
  - conf\_hypergfg, 725
  - conf\_hypergl, 725
  - hyperg, 725
  - hypergfg, 726
  - hypergl, 726
  - norm, 726
  - polar, 726
- std::tr1::\_\_detail, 727
- std::tr2, 727
  - \_\_detail, 728
- std::tr2::\_\_dynamic\_bitset\_base< \_WordT, \_Alloc >, 762
  - \_M\_w, 764
- std::tr2::\_\_reflection\_typelist< \_Elements >, 799
- std::tr2::\_\_reflection\_typelist< \_First, \_Rest... >, 799
- std::tr2::\_\_reflection\_typelist<>, 800
- std::tr2::bases< \_Tp >, 1088

- std::tr2::bool\_set, 1770
  - bool\_set, 1771
  - equals, 1771
  - is\_emptyset, 1771
  - is\_indeterminate, 1771
  - is\_singleton, 1771
  - operator bool, 1771
- std::tr2::direct\_bases< \_Tp >, 2051
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2069
  - all, 2074
  - any, 2074
  - append, 2074, 2075
  - clear, 2075
  - count, 2075
  - dynamic\_bitset, 2073, 2074
  - empty, 2075
  - find\_first, 2075
  - find\_next, 2075
  - flip, 2076
  - get\_allocator, 2076
  - max\_size, 2076
  - none, 2076
  - num\_blocks, 2076
  - operator<<, 2077
  - operator<=, 2077
  - operator>>, 2078
  - operator>=, 2078
  - operator=, 2077
  - operator=, 2078
  - operator&=, 2077
  - operator[], 2078, 2079
  - operator~, 2080
  - operator^=, 2079
  - operator|=, 2079
  - push\_back, 2080
  - reset, 2080
  - resize, 2080
  - set, 2080, 2081
  - size, 2081
  - swap, 2081
  - test, 2081
  - to\_string, 2082
  - to\_ullong, 2082
  - to\_ulong, 2082
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >::reference, 2812
- std::try\_to\_lock\_t, 3122
- std::tuple< \_Elements >, 3122
- std::tuple< \_T1, \_T2 >, 3124
- std::tuple\_element< 0, pair< \_Tp1, \_Tp2 > >, 3127
- std::tuple\_element< 1, pair< \_Tp1, \_Tp2 > >, 3127
- std::tuple\_element< \_\_i, \_Tp >, 3126
- std::tuple\_element< \_\_i, tuple< \_Types... > >, 3127
- std::tuple\_element< \_Ind, array< \_Tp, \_Nm > >, 3127
- std::tuple\_size< \_Tp >, 3128
- std::tuple\_size< array< \_Tp, \_Nm > >, 3128
- std::tuple\_size< pair< \_Tp1, \_Tp2 > >, 3129
- std::tuple\_size< tuple< \_Elements... > >, 3130
- std::type\_index, 3131
- std::type\_info, 3132
  - ~type\_info, 3133
  - before, 3133
  - name, 3133
- std::unary\_function< \_Arg, \_Result >, 3138
  - argument\_type, 3138
  - result\_type, 3138
- std::unary\_negate< \_Predicate >, 3139
  - argument\_type, 3139
  - result\_type, 3139
- std::underflow\_error, 3141
  - what, 3141
- std::underlying\_type< \_Tp >, 3142
- std::uniform\_int\_distribution< \_IntType >, 3142
  - max, 3143
  - min, 3143
  - operator(), 3143
  - operator==, 3144
  - param, 3144
  - reset, 3144
  - result\_type, 3143
  - uniform\_int\_distribution, 3143
- std::uniform\_int\_distribution< \_IntType >::param\_type, 2728
- std::uniform\_real\_distribution< \_RealType >, 3144
  - max, 3146
  - min, 3146
  - operator(), 3146
  - operator==, 3147
  - param, 3146
  - reset, 3147
  - result\_type, 3145
  - uniform\_real\_distribution, 3145
- std::uniform\_real\_distribution< \_RealType >::param\_type, 2729
- std::unique\_lock< \_Mutex >, 3147
  - swap, 3148
- std::unique\_ptr< \_Tp, \_Dp >, 3148
  - ~unique\_ptr, 3151
  - get, 3151
  - get\_deleter, 3152
  - operator bool, 3152
  - operator->, 3152
  - operator=, 3152
  - operator\*, 3152
  - release, 3153
  - reset, 3153
  - swap, 3153
  - unique\_ptr, 3150, 3151

- std::unique\_ptr< \_Tp[], \_Dp >, 3153
  - ~unique\_ptr, 3157, 3159
  - get, 3159
  - get\_deleter, 3159
  - operator bool, 3159
  - operator<<, 3162
  - operator->, 3160
  - operator=, 3160, 3161
  - operator[], 3161
  - operator\*, 3160
  - release, 3161
  - reset, 3161, 3162
  - swap, 3162, 3163
  - unique\_ptr, 3156–3158
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3166
  - allocator\_type, 3170
  - at, 3174
  - begin, 3175
  - bucket\_count, 3176
  - cbegin, 3176
  - cend, 3176
  - clear, 3177
  - const\_iterator, 3170
  - const\_local\_iterator, 3170
  - const\_pointer, 3170
  - const\_reference, 3170
  - contains, 3177
  - count, 3178
  - difference\_type, 3170
  - emplace, 3178
  - emplace\_hint, 3180
  - empty, 3180
  - end, 3180, 3181
  - equal\_range, 3181, 3182
  - erase, 3183, 3184
  - find, 3184, 3185
  - get\_allocator, 3186
  - hash\_function, 3186
  - hasher, 3171
  - insert, 3186–3189
  - iterator, 3171
  - key\_eq, 3189
  - key\_equal, 3171
  - key\_type, 3171
  - load\_factor, 3189
  - local\_iterator, 3171
  - mapped\_type, 3171
  - max\_bucket\_count, 3190
  - max\_load\_factor, 3190
  - max\_size, 3190
  - operator=, 3190, 3191
  - operator[], 3191
  - pointer, 3171
  - reference, 3171
  - rehash, 3192
  - reserve, 3192
  - size, 3192
  - size\_type, 3172
  - swap, 3192
  - unordered\_map, 3172, 3173
  - value\_type, 3172
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3197
  - allocator\_type, 3200
  - begin, 3205
  - bucket\_count, 3206
  - cbegin, 3206
  - cend, 3206
  - clear, 3207
  - const\_iterator, 3200
  - const\_local\_iterator, 3201
  - const\_pointer, 3201
  - const\_reference, 3201
  - contains, 3207
  - count, 3208
  - difference\_type, 3201
  - emplace, 3208
  - emplace\_hint, 3209
  - empty, 3209
  - end, 3209, 3210
  - equal\_range, 3210, 3211
  - erase, 3212, 3213
  - find, 3213, 3214
  - get\_allocator, 3215
  - hash\_function, 3215
  - hasher, 3201
  - insert, 3215, 3216, 3218
  - iterator, 3201
  - key\_eq, 3219
  - key\_equal, 3201
  - key\_type, 3202
  - load\_factor, 3219
  - local\_iterator, 3202
  - mapped\_type, 3202
  - max\_bucket\_count, 3219
  - max\_load\_factor, 3219
  - max\_size, 3221
  - operator=, 3221
  - pointer, 3202
  - reference, 3202
  - rehash, 3221
  - reserve, 3222
  - size, 3222
  - size\_type, 3202
  - swap, 3222
  - unordered\_multimap, 3203, 3204
  - value\_type, 3202

- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc  
>, 3225
  - allocator\_type, 3229
  - begin, 3233
  - bucket\_count, 3234
  - cbegin, 3234
  - cend, 3234
  - clear, 3235
  - const\_iterator, 3229
  - const\_local\_iterator, 3229
  - const\_pointer, 3229
  - const\_reference, 3229
  - contains, 3235
  - count, 3236
  - difference\_type, 3229
  - emplace, 3236
  - emplace\_hint, 3237
  - empty, 3237
  - end, 3237, 3238
  - equal\_range, 3238, 3239
  - erase, 3239–3241
  - find, 3241, 3243
  - get\_allocator, 3243
  - hash\_function, 3244
  - hasher, 3230
  - insert, 3244–3246
  - iterator, 3230
  - key\_eq, 3246
  - key\_equal, 3230
  - key\_type, 3230
  - load\_factor, 3246
  - local\_iterator, 3230
  - max\_bucket\_count, 3246
  - max\_load\_factor, 3246, 3247
  - max\_size, 3247
  - operator=, 3247, 3248
  - pointer, 3230
  - reference, 3230
  - rehash, 3248
  - reserve, 3248
  - size, 3248
  - size\_type, 3231
  - swap, 3248
  - unordered\_multiset, 3231, 3232
  - value\_type, 3231
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3252
  - allocator\_type, 3255
  - begin, 3259, 3261
  - bucket\_count, 3261
  - cbegin, 3261
  - cend, 3262
  - clear, 3262
  - const\_iterator, 3255
  - const\_local\_iterator, 3255
  - const\_pointer, 3255
  - const\_reference, 3256
  - contains, 3262
  - count, 3263
  - difference\_type, 3256
  - emplace, 3263
  - emplace\_hint, 3264
  - empty, 3264
  - end, 3264, 3265
  - equal\_range, 3265, 3267
  - erase, 3269, 3270
  - find, 3270, 3271
  - get\_allocator, 3272
  - hash\_function, 3272
  - hasher, 3256
  - insert, 3272–3274
  - iterator, 3256
  - key\_eq, 3274
  - key\_equal, 3256
  - key\_type, 3256
  - load\_factor, 3274
  - local\_iterator, 3256
  - max\_bucket\_count, 3275
  - max\_load\_factor, 3275
  - max\_size, 3275
  - operator=, 3275, 3276
  - pointer, 3257
  - reference, 3257
  - rehash, 3276
  - reserve, 3276
  - size, 3277
  - size\_type, 3257
  - swap, 3277
  - unordered\_set, 3257–3259
  - value\_type, 3257
- std::uses\_allocator< tuple< \_Types... >, \_Alloc >, 3279
- std::uses\_allocator< typename, typename >, 3279
- std::valarray< \_Tp >, 3280
  - operator!, 3282
  - operator<=, 3284
  - operator>=, 3284
  - operator+, 3283
  - operator+=, 3283
  - operator-, 3283
  - operator-=, 3283
  - operator/=, 3284
  - operator%=, 3282
  - operator&=, 3282
  - operator\*=, 3283
  - operator~, 3285
  - operator^=, 3284
  - operator|=, 3285
  - valarray, 3282



std::vector< \_Tp, \_Alloc >, 3288  
   \_M\_allocate\_and\_copy, 3294  
   \_M\_range\_check, 3294  
   ~vector, 3294  
   assign, 3294, 3295  
   at, 3295, 3296  
   back, 3296  
   begin, 3297  
   capacity, 3297  
   cbegin, 3297  
   cend, 3297  
   clear, 3297  
   crbegin, 3298  
   crend, 3298  
   data, 3298  
   emplace, 3298  
   empty, 3298  
   end, 3298, 3299  
   erase, 3299  
   front, 3300  
   get\_allocator, 3300  
   insert, 3300–3302  
   max\_size, 3302  
   operator=, 3302, 3303  
   operator[], 3303  
   pop\_back, 3304  
   push\_back, 3304  
   rbegin, 3304  
   rend, 3305  
   reserve, 3305  
   resize, 3305, 3306  
   shrink\_to\_fit, 3306  
   size, 3306  
   swap, 3306  
   vector, 3292–3294  
 std::vector< bool, \_Alloc >, 3307  
   \_M\_allocate\_and\_copy, 3315  
   \_M\_range\_check, 3315  
   ~vector, 3315  
   assign, 3315, 3317  
   at, 3317  
   back, 3318  
   begin, 3318  
   capacity, 3318  
   cbegin, 3318  
   cend, 3318  
   clear, 3318  
   crbegin, 3319  
   crend, 3319  
   data, 3319  
   emplace, 3319  
   empty, 3319  
   end, 3319  
   erase, 3320  
   front, 3320  
   get\_allocator, 3320, 3321  
   insert, 3321, 3322  
   max\_size, 3322  
   operator=, 3323  
   operator[], 3323, 3324  
   pop\_back, 3324  
   push\_back, 3324  
   rbegin, 3324, 3325  
   rend, 3325  
   reserve, 3325  
   resize, 3325, 3326  
   shrink\_to\_fit, 3326  
   size, 3326  
   swap, 3326  
   vector, 3312–3314  
 std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3328  
   \_M\_buf\_locale, 3343  
   \_M\_in\_beg, 3343  
   \_M\_in\_cur, 3343  
   \_M\_in\_end, 3343  
   \_M\_out\_beg, 3343  
   \_M\_out\_cur, 3343  
   \_M\_out\_end, 3343  
   \_\_streambuf\_type, 3330  
   char\_type, 3330  
   eback, 3331  
   egptr, 3331  
   epptr, 3331  
   gbump, 3332  
   getloc, 3332  
   gptr, 3332  
   imbue, 3333  
   in\_avail, 3333  
   int\_type, 3330  
   off\_type, 3330  
   overflow, 3333  
   pbackfail, 3334  
   pbase, 3334  
   pbump, 3334  
   pos\_type, 3330  
   pptr, 3335  
   pubimbue, 3335  
   pubseekoff, 3335  
   pubseekpos, 3336  
   pubsetbuf, 3336  
   pubsync, 3336  
   sbumpc, 3336  
   seekoff, 3336  
   seekpos, 3337  
   setbuf, 3337  
   setg, 3337  
   setp, 3338  
   sgetc, 3338

- sgetn, 3338
- showmanyc, 3339
- snextc, 3339
- sputbackc, 3339
- sputc, 3340
- sputn, 3340
- state, 3341
- sungetc, 3341
- sync, 3341
- traits\_type, 3330
- uflow, 3341
- underflow, 3341
- wbuffer\_convert, 3330, 3331
- xsggetn, 3342
- xspn, 3342
- std::weak\_ptr< \_Tp >, 3344
- std::weibull\_distribution< \_RealType >, 3345
  - a, 3346
  - b, 3346
  - max, 3346
  - min, 3346
  - operator(), 3346
  - operator==, 3347
  - param, 3347
  - reset, 3347
  - result\_type, 3346
- std::weibull\_distribution< \_RealType >::param\_type, 2729
- std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, 3347
  - converted, 3349
  - from\_bytes, 3349, 3350
  - state, 3350
  - to\_bytes, 3350, 3351
  - wstring\_convert, 3348, 3349
- std\_abs.h, 3465
- std\_function.h, 3465
- std\_mutex.h, 3466
- std\_thread.h, 3467
- stdatomic.h, 3755
- stdbit.h, 3755
- stdc++.h, 3800
- stdckdint.h, 3755
- stddev
  - std::normal\_distribution< \_RealType >, 2537
- stdexcept, 3755
- stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 2990
- stdio\_filebuf.h, 3671
- stdio\_sync\_filebuf.h, 3671
- stdlib.h, 3756
  - abs, 3756
- stdtr1c++.h, 3800
- stl\_algo.h, 3467
- stl\_algobase.h, 3476
- stl\_bvector.h, 3481
- stl\_construct.h, 3482
- stl\_deque.h, 3483
  - \_GLIBCXX\_DEQUE\_BUF\_SIZE, 3483
- stl\_function.h, 3484
- stl\_heap.h, 3485
- stl\_iterator.h, 3487, 3489
- stl\_iterator\_base\_funcs.h, 3490
- stl\_iterator\_base\_types.h, 3491
- stl\_list.h, 3492
- stl\_map.h, 3493
- stl\_multimap.h, 3494
- stl\_multiset.h, 3495
- stl\_numeric.h, 3496
- stl\_pair.h, 3496
- stl\_queue.h, 3497
- stl\_raw\_storage\_iter.h, 3499
- stl\_relops.h, 3499
- stl\_set.h, 3499
- stl\_stack.h, 3500
- stl\_tempbuf.h, 3501
- stl\_tree.h, 3502
- stl\_uninitialized.h, 3502
- stl\_vector.h, 3503
- stop\_token, 3756
- str
  - std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 1340
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1437
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1664, 1665
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1707, 1708
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2415
  - std::sub\_match< \_Biter >, 3036
- stream\_iterator.h, 3504
- streambuf, 3756
  - I/O, 165
- streambuf.tcc, 3504
- streambuf\_iterator.h, 3504
- streambuf\_type
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 2295
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 2691
- streamoff
  - std, 580
- streampos
  - std, 580
- streamsize
  - std, 580
- stride
  - Numeric Arrays, 232
- string, 3757, 3759, 3760

- Strings, 273
- string\_conversions.h, 3672
- string\_type
  - std::\_\_cxx11::collate< \_CharT >, 1857
  - std::\_\_cxx11::collate\_byname< \_CharT >, 1862
  - std::messages< \_CharT >, 2430
  - std::money\_get< \_CharT, \_InIter >, 2441
  - std::money\_put< \_CharT, \_OutIter >, 2445
  - std::moneypunct< \_CharT, \_Intl >, 2450
  - std::numpunct< \_CharT >, 2673
- string\_view, 3760, 3762
- string\_view.tcc, 3505
- stringbuf
  - I/O, 165
- stringfwd.h, 3506
- Strings, 273
  - string, 273
  - u16string, 273
  - u32string, 273
  - wstring, 273
- stringstream
  - I/O, 166
- strstream, 3362
- substr
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 857
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1544
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1642, 1643
- subtract\_with\_carry\_engine
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3039
- subtractive\_rng
  - \_\_gnu\_cxx::subtractive\_rng, 3043
- suffix
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2416
- sum
  - Numeric Arrays, 232
- sungetc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2097
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3004
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3022
  - std::basic\_filebuf< \_CharT, \_Traits >, 1105
  - std::basic\_streambuf< \_CharT, \_Traits >, 1470
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 1665
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3341
- swap
  - \_\_gnu\_cxx, 403
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 857
  - \_\_gnu\_cxx::pair< \_T1, \_T2 >, 2710
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1544
  - \_\_gnu\_parallel::\_iteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 896, 898
  - \_\_gnu\_pbds::sample\_probe\_fn, 2844
  - \_\_gnu\_pbds::sample\_range\_hashing, 2845
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 2846
  - \_\_gnu\_pbds::sample\_resize\_policy, 2849
  - \_\_gnu\_pbds::sample\_resize\_trigger, 2852
  - \_\_gnu\_pbds::sample\_size\_policy, 2853
  - \_\_gnu\_pbds::sample\_update\_policy, 2857
  - Futures, 115
  - Metaprogramming, 353
  - Numeric Arrays, 232
  - Pointer Abstractions, 342
  - Regular Expressions, 271
  - std, 642–644
  - std::\_\_debug, 654
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1455
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1643
  - std::copyable\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 1908
  - std::deque< \_Tp, \_Alloc >, 2050
  - std::experimental::fundamentals\_v1::any, 1066
  - std::forward\_list< \_Tp, \_Alloc >, 2157
  - std::function< \_Res(\_ArgTypes...)>, 2169
  - std::list< \_Tp, \_Alloc >, 2346
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2402
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2416
  - std::move\_only\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>, 2471
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2496
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2521
  - std::pair< \_T1, \_T2 >, 2714
  - std::set< \_Key, \_Compare, \_Alloc >, 2949
  - std::shared\_ptr< \_Tp >, 2967
  - std::sub\_match< \_Biter >, 3036
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2081
  - std::unique\_lock< \_Mutex >, 3148
  - std::unique\_ptr< \_Tp, \_Dp >, 3153
  - std::unique\_ptr< \_Tp[], \_Dp >, 3162, 3163
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3192
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3222
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3248
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3277
  - std::vector< \_Tp, \_Alloc >, 3306
  - std::vector< bool, \_Alloc >, 3326
  - Type-safe container of any type, 290
  - Utilities, 304, 305

- swap\_ranges
  - Mutating, [37](#)
- sync
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2097](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3004](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3022](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1105](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1151](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1194](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1263](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1301](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2890](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1340](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1470](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1665](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1708](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3341](#)
- sync\_with\_stdio
  - std::basic\_fstream< \_CharT, \_Traits >, [1151](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1194](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1218](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1263](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1301](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2890](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1340](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1373](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1405](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2918](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1437](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1708](#)
  - std::ios\_base, [2264](#)
- syncstream, [3763](#)
- syntax\_option\_type
  - std::regex\_constants, [710](#)
- synth\_access\_traits
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [3120](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [3122](#)
- synth\_access\_traits.hpp, [3644](#)
- system\_category
  - Diagnostics, [126](#)
- system\_error, [3763](#), [3764](#)
- t
  - std::binomial\_distribution< \_IntType >, [1750](#)
- table
  - std::ctype< char >, [1951](#)
  - std::ctype\_byname< char >, [2008](#)
- table\_size
  - std::ctype< char >, [1958](#)
  - std::ctype\_byname< char >, [2013](#)
- tag\_and\_trait.hpp, [3655](#)
- Tags, [144](#)
  - trivial\_iterator\_difference\_type, [144](#)
- tags.h, [3745](#)
- tan
  - Complex Numbers, [187](#)
  - math.h, [3702](#)
- tanh
  - Complex Numbers, [187](#)
  - math.h, [3702](#)
- target
  - std::function< \_Res(\_ArgTypes...)>, [2170](#)
- target\_type
  - std::function< \_Res(\_ArgTypes...)>, [2170](#)
- Technical Specifications, [274](#)
- tellg
  - std::basic\_fstream< \_CharT, \_Traits >, [1151](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1195](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1263](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1301](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2890](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1341](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1709](#)
- tellp
  - std::basic\_fstream< \_CharT, \_Traits >, [1152](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1264](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1375](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1405](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2918](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1439](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1709](#)
- temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, [3047](#)
- terminate
  - Exceptions, [131](#)
- terminate\_handler
  - Exceptions, [128](#)
- test
  - std::bitset< \_Nb >, [1769](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2081](#)
- text\_encoding, [3764](#)
- text\_encoding-data.h, [3506](#)
- tgmath.h, [3764](#)

- thin\_heap\_.hpp, 3649
- this\_thread\_sleep.h, 3506
- thousands\_sep
  - std::moneypunct< \_CharT, \_Intl >, 2457
  - std::moneypunct\_byname< \_CharT, \_Intl >, 2464
  - std::numpunct< \_CharT >, 2676
  - std::numpunct\_byname< \_CharT >, 2680
- thread, 3765
- Threads, 119
- throw\_allocator.h, 3672
- throw\_with\_nested
  - Exceptions, 131
- tie
  - std::basic\_fstream< \_CharT, \_Traits >, 1152
  - std::basic\_ifstream< \_CharT, \_Traits >, 1195
  - std::basic\_ios< \_CharT, \_Traits >, 1219
  - std::basic\_iostream< \_CharT, \_Traits >, 1264
  - std::basic\_istream< \_CharT, \_Traits >, 1302
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2891
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1341
  - std::basic\_ofstream< \_CharT, \_Traits >, 1375
  - std::basic\_ostream< \_CharT, \_Traits >, 1405
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2918
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1439
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1709
  - Utilities, 305
- Time, 356
  - abs, 366
  - ceil, 366, 367
  - clock\_cast, 367
  - days, 365
  - duration\_cast, 367
  - floor, 368
  - high\_resolution\_clock, 365
  - hours, 365
  - local\_time\_format, 369
  - microseconds, 365
  - milliseconds, 365
  - minutes, 365
  - months, 365
  - nanoseconds, 365
  - operator<, 373
  - operator<<, 373
  - operator<=, 374
  - operator<=>, 374
  - operator>, 374
  - operator>=, 374
  - operator+, 372
  - operator-, 372
  - operator/, 373
  - operator==, 374
  - operator%, 371
  - operator\*, 371
  - round, 374, 376
  - seconds, 365
  - time\_point\_cast, 376
  - weeks, 366
  - years, 366
- time
  - std::locale, 2357
- time\_get
  - std::time\_get< \_CharT, \_InIter >, 3063
- time\_members.h, 3800
- time\_point\_cast
  - Time, 376
- time\_put
  - std::time\_put< \_CharT, \_OutIter >, 3085
- tinytness\_before
  - std::\_\_numeric\_limits\_base, 790
  - std::numeric\_limits< \_Tp >, 2577
  - std::numeric\_limits< bool >, 2582
  - std::numeric\_limits< char >, 2587
  - std::numeric\_limits< char16\_t >, 2592
  - std::numeric\_limits< char32\_t >, 2598
  - std::numeric\_limits< double >, 2603
  - std::numeric\_limits< float >, 2608
  - std::numeric\_limits< int >, 2613
  - std::numeric\_limits< long >, 2619
  - std::numeric\_limits< long double >, 2624
  - std::numeric\_limits< long long >, 2629
  - std::numeric\_limits< short >, 2634
  - std::numeric\_limits< signed char >, 2640
  - std::numeric\_limits< unsigned char >, 2645
  - std::numeric\_limits< unsigned int >, 2650
  - std::numeric\_limits< unsigned long >, 2655
  - std::numeric\_limits< unsigned long long >, 2661
  - std::numeric\_limits< unsigned short >, 2666
  - std::numeric\_limits< wchar\_t >, 2671
- TLB\_size
  - \_\_gnu\_parallel::Settings, 1015
- to\_address
  - Pointer Abstractions, 342, 343
- to\_array
  - Array creation functions, 282
- to\_bytes
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, 3350, 3351
- to\_string
  - std::bitset< \_Nb >, 1769
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2082
- to\_ulong
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2082
- to\_ulong
  - std::bitset< \_Nb >, 1770
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2082

- Todo List, [2](#)
- tolower
  - std, [645](#)
  - std::\_\_ctype\_abstract\_base< \_CharT >, [759](#), [760](#)
  - std::ctype< \_CharT >, [1920](#), [1921](#)
  - std::ctype< char >, [1951–1953](#)
  - std::ctype< wchar\_t >, [1974](#)
  - std::ctype\_byname< \_CharT >, [1988](#)
  - std::ctype\_byname< char >, [2008](#), [2009](#)
- top
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [2768](#)
  - std::stack< \_Tp, \_Sequence >, [2986](#), [2987](#)
- toupper
  - std, [645](#)
  - std::\_\_ctype\_abstract\_base< \_CharT >, [760](#)
  - std::ctype< \_CharT >, [1921](#)
  - std::ctype< char >, [1953–1955](#)
  - std::ctype< wchar\_t >, [1975](#)
  - std::ctype\_byname< \_CharT >, [1988](#), [1989](#)
  - std::ctype\_byname< char >, [2010](#), [2011](#)
- TR1 Mathematical Special Functions, [246](#)
  - assoc\_laguerre, [248](#)
  - assoc\_legendre, [248](#)
  - beta, [249](#)
  - comp\_ellint\_1, [249](#)
  - comp\_ellint\_2, [249](#)
  - comp\_ellint\_3, [249](#)
  - cyl\_bessel\_i, [249](#)
  - cyl\_bessel\_j, [249](#)
  - cyl\_bessel\_k, [249](#)
  - cyl\_neumann, [250](#)
  - ellint\_1, [250](#)
  - ellint\_2, [250](#)
  - ellint\_3, [250](#)
  - expint, [250](#)
  - hermite, [250](#)
  - laguerre, [250](#)
  - legendre, [251](#)
  - riemann\_zeta, [251](#)
  - sph\_bessel, [251](#)
  - sph\_legendre, [251](#)
  - sph\_neumann, [251](#)
- trace\_fn\_imps.hpp, [3627](#)
- Traits, [146](#)
- traits.hpp, [3612](#), [3613](#)
- traits\_type
  - std::basic\_ios< \_CharT, \_Traits >, [1209](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2869](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1460](#)
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [2295](#)
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [2687](#)
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [2691](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3330](#)
- transform
  - Mutating, [37](#), [38](#)
  - std::\_\_cxx11::collate< \_CharT >, [1860](#)
  - std::regex\_traits< \_Ch\_type >, [2826](#)
- transform\_exclusive\_scan
  - Generalized Numeric operations, [15](#)
- transform\_inclusive\_scan
  - Generalized Numeric operations, [15](#), [16](#)
- transform\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1015](#)
- transform\_primary
  - std::regex\_traits< \_Ch\_type >, [2827](#)
- transform\_reduce
  - Generalized Numeric operations, [17](#), [18](#)
- translate
  - std::regex\_traits< \_Ch\_type >, [2827](#)
- translate\_nocase
  - std::regex\_traits< \_Ch\_type >, [2827](#)
- traps
  - std::\_\_numeric\_limits\_base, [790](#)
  - std::numeric\_limits< \_Tp >, [2577](#)
  - std::numeric\_limits< bool >, [2582](#)
  - std::numeric\_limits< char >, [2587](#)
  - std::numeric\_limits< char16\_t >, [2592](#)
  - std::numeric\_limits< char32\_t >, [2598](#)
  - std::numeric\_limits< double >, [2603](#)
  - std::numeric\_limits< float >, [2608](#)
  - std::numeric\_limits< int >, [2613](#)
  - std::numeric\_limits< long >, [2619](#)
  - std::numeric\_limits< long double >, [2624](#)
  - std::numeric\_limits< long long >, [2629](#)
  - std::numeric\_limits< short >, [2634](#)
  - std::numeric\_limits< signed char >, [2640](#)
  - std::numeric\_limits< unsigned char >, [2645](#)
  - std::numeric\_limits< unsigned int >, [2650](#)
  - std::numeric\_limits< unsigned long >, [2655](#)
  - std::numeric\_limits< unsigned long long >, [2661](#)
  - std::numeric\_limits< unsigned short >, [2666](#)
  - std::numeric\_limits< wchar\_t >, [2671](#)
- tree
  - \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >, [3094](#)
- tree\_policy.hpp, [3656](#)
- tree\_trace\_base.hpp, [3650](#)
- trie
  - \_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >, [3107](#)
- trie\_policy.hpp, [3656](#)
- trie\_policy\_base.hpp, [3651](#)
- trie\_string\_access\_traits\_imp.hpp, [3651](#)
- trivial\_iterator\_difference\_type
  - Tags, [144](#)
- true\_type
  - Metaprogramming, [352](#)

truename  
     std::num\_punct<\_CharT>, [2676](#)  
     std::num\_punct\_byname<\_CharT>, [2681](#)  
 trunc  
     std::basic\_fstream<\_CharT, \_Traits>, [1159](#)  
     std::basic\_ifstream<\_CharT, \_Traits>, [1202](#)  
     std::basic\_ios<\_CharT, \_Traits>, [1225](#)  
     std::basic\_iostream<\_CharT, \_Traits>, [1271](#)  
     std::basic\_istream<\_CharT, \_Traits>, [1308](#)  
     std::basic\_istream<\_CharT, \_Traits>::sentry, [2896](#)  
     std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1348](#)  
     std::basic\_ofstream<\_CharT, \_Traits>, [1381](#)  
     std::basic\_ostream<\_CharT, \_Traits>, [1412](#)  
     std::basic\_ostream<\_CharT, \_Traits>::sentry, [2924](#)  
     std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1445](#)  
     std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1716](#)  
     std::ios\_base, [2270](#)  
 try\_lock  
     Mutexes, [117](#)  
 try\_to\_lock  
     Mutexes, [118](#)  
 tuple, [3767](#), [3768](#)  
 tuple\_cat  
     Utilities, [306](#)  
 type  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, Mapped, \_Alloc, cc\_hash\_tag, Policy\_TI>, [1890](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_TI>, [1891](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, Mapped, \_Alloc, list\_update\_tag, Policy\_TI>, [1891](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, Mapped, \_Alloc, ov\_tree\_tag, Policy\_TI>, [1892](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, Mapped, \_Alloc, rb\_tree\_tag, Policy\_TI>, [1892](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, Mapped, \_Alloc, splay\_tree\_tag, Policy\_TI>, [1893](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, null\_type, \_Alloc, cc\_hash\_tag, Policy\_TI>, [1893](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, null\_type, \_Alloc, gp\_hash\_tag, Policy\_TI>, [1894](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, null\_type, \_Alloc, list\_update\_tag, Policy\_TI>, [1894](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_TI>, [1895](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_TI>, [1895](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_TI>, [1896](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type>, [1888](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, binomial\_heap\_tag, null\_type>, [1888](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type>, [1889](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type>, [1889](#)  
     \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type>, [1890](#)  
     \_\_gnu\_pbds::detail::default\_comb\_hash\_fn, [2019](#)  
     \_\_gnu\_pbds::detail::default\_eq\_fn<Key>, [2022](#)  
     \_\_gnu\_pbds::detail::default\_hash\_fn<Key>, [2022](#)  
     \_\_gnu\_pbds::detail::default\_probe\_fn<Comb\_Probe\_Fn>, [2024](#)  
     \_\_gnu\_pbds::detail::default\_resize\_policy<Comb\_Hash\_Fn>, [2024](#)  
     \_\_gnu\_pbds::detail::default\_trie\_access\_traits<std::basic\_string<Char, Char\_Traits, std::allocator<char>>>, [2025](#)  
     \_\_gnu\_pbds::detail::default\_update\_policy, [2026](#)  
     \_\_gnu\_pbds::detail::entry\_cmp<\_VTp, Cmp\_Fn, \_Alloc, true>, [2106](#)  
     Metaprogramming, [352](#)  
     std::experimental::fundamentals\_v1::any, [1066](#)  
 Type-safe container of any type, [287](#)  
     any\_cast, [288–290](#)  
     swap, [290](#)  
 type\_traits, [3770](#), [3773](#), [3774](#)  
 type\_traits.h, [3673](#)  
 type\_utils.hpp, [3651](#)  
 typeindex, [3778](#)  
 typeinfo, [3805](#)  
 typelist.h, [3673](#)  
 types.h, [3746](#)  
 types\_traits.hpp, [3652](#)  
 u16streampos  
     std, [580](#)  
 u16string  
     \_\_gnu\_debug, [411](#)



- Strings, [273](#)
- u32streampos
  - std, [580](#)
- u32string
  - \_\_gnu\_debug, [411](#)
  - Strings, [273](#)
- u8path
  - Filesystem, [162](#), [163](#)
- uflow
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2098](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3004](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3023](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1106](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1471](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1665](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3341](#)
- uncaught\_exception
  - Exceptions, [131](#)
- undecare\_no\_pointers
  - Pointer Safety and Garbage Collection, [344](#)
- undecare\_reachable
  - Pointer Safety and Garbage Collection, [344](#)
- underflow
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2098](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3004](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3023](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1106](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1471](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1666](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3341](#)
- underlying\_type\_t
  - Metaprogramming, [352](#)
- unexpected
  - Exceptions, [132](#)
- unexpected\_handler
  - Exceptions, [128](#)
- unget
  - std::basic\_fstream< \_CharT, \_Traits >, [1153](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1196](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1265](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1302](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2891](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1342](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1710](#)
- unicode-data.h, [3507](#)
- unicode.h, [3507](#)
- Uniform Distributions, [242](#)
  - operator<<, [243](#)
  - operator>>, [243](#), [244](#)
- uniform\_int\_dist.h, [3508](#)
- uniform\_int\_distribution
  - std::uniform\_int\_distribution< \_IntType >, [3143](#)
- uniform\_real\_distribution
  - std::uniform\_real\_distribution< \_RealType >, [3145](#)
- uninitialized\_copy
  - Memory, [319](#)
- uninitialized\_copy\_n
  - Memory, [319](#)
  - SGL, [156](#)
- uninitialized\_default\_construct
  - Memory, [320](#)
- uninitialized\_default\_construct\_n
  - Memory, [320](#)
- uninitialized\_fill
  - Memory, [320](#)
- uninitialized\_fill\_n
  - Memory, [321](#)
- uninitialized\_move
  - Memory, [321](#)
- uninitialized\_move\_n
  - Memory, [322](#)
- uninitialized\_value\_construct
  - Memory, [322](#)
- uninitialized\_value\_construct\_n
  - Memory, [322](#)
- unique
  - Mutating, [38](#), [40](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2158](#)
  - std::list< \_Tp, \_Alloc >, [2346](#)
  - std::shared\_ptr< \_Tp >, [2967](#)
- unique\_copy
  - Mutating, [40](#), [41](#)
- unique\_copy.h, [3746](#)
- unique\_copy\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1015](#)
- unique\_lock.h, [3509](#)
- unique\_ptr
  - std::unique\_ptr< \_Tp, \_Dp >, [3150](#), [3151](#)
  - std::unique\_ptr< \_Tp[], \_Dp >, [3156–3158](#)
- unique\_ptr.h, [3509](#)
- unitbuf
  - std, [645](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1159](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1202](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1225](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1271](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1308](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2896](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1348](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1382](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1412](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2924](#)



- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1446
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1716
- std::ios\_base, 2270
- Unordered Associative, 122
- unordered\_map, 3778, 3780, 3781
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3172, 3173
- unordered\_map.h, 3510
- unordered\_multimap
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3203, 3204
- unordered\_multiset
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3231, 3232
- unordered\_set, 3781, 3783, 3784
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3257–3259
- unordered\_set.h, 3512
- unsetf
  - std::basic\_fstream< \_CharT, \_Traits >, 1153
  - std::basic\_ifstream< \_CharT, \_Traits >, 1196
  - std::basic\_ios< \_CharT, \_Traits >, 1219
  - std::basic\_iostream< \_CharT, \_Traits >, 1265
  - std::basic\_istream< \_CharT, \_Traits >, 1303
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2891
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1342
  - std::basic\_ofstream< \_CharT, \_Traits >, 1375
  - std::basic\_ostream< \_CharT, \_Traits >, 1406
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2919
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1439
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1710
  - std::ios\_base, 2265
- unshift
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, 745
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 1804
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 1812, 1813
  - std::codecvt< char, char, mbstate\_t >, 1821, 1822
  - std::codecvt< char16\_t, char, mbstate\_t >, 1830
  - std::codecvt< char32\_t, char, mbstate\_t >, 1839
  - std::codecvt< wchar\_t, char, mbstate\_t >, 1848, 1849
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 1855
- update\_fn\_imps.hpp, 3644
- upper\_bound
  - Binary Search, 85
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2402, 2403
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2496, 2497
- std::multiset< \_Key, \_Compare, \_Alloc >, 2521, 2522
- std::set< \_Key, \_Compare, \_Alloc >, 2949, 2950
- uppercase
  - std, 645
  - std::basic\_fstream< \_CharT, \_Traits >, 1159
  - std::basic\_ifstream< \_CharT, \_Traits >, 1202
  - std::basic\_ios< \_CharT, \_Traits >, 1225
  - std::basic\_iostream< \_CharT, \_Traits >, 1271
  - std::basic\_istream< \_CharT, \_Traits >, 1308
  - std::basic\_istream< \_CharT, \_Traits >::sentry, 2896
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1348
  - std::basic\_ofstream< \_CharT, \_Traits >, 1382
  - std::basic\_ostream< \_CharT, \_Traits >, 1412
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2924
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1446
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 1716
  - std::ios\_base, 2270
- use\_count
  - std::shared\_ptr< \_Tp >, 2967
- use\_facet
  - Locales, 175
  - std::locale, 2355
  - std::locale::id, 2239
- uses\_allocator\_args.h, 3515
- Utilities, 295
  - \_\_addressof, 298
  - \_\_invoke, 299
  - addressof, 299
  - declval, 299
  - forward, 300
  - forward\_as\_tuple, 301
  - get, 301
  - make\_pair, 301
  - make\_tuple, 302
  - move, 302
  - move\_if\_noexcept, 303
  - operator<=>, 304
  - operator==, 304
  - pair, 298
  - piecewise\_construct, 306
  - swap, 304, 305
  - tie, 305
  - tuple\_cat, 306
- utility, 3785
- utility.h, 3515
- valarray, 3785
  - Numeric Arrays, 220, 221

- std::valarray< \_Tp >, 3282
- valarray\_after.h, 3516
- valarray\_array.h, 3526
- valarray\_array.tcc, 3534
- valarray\_before.h, 3534
- valid\_prefix
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 937
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 939
- value
  - std::error\_code, 2114
  - std::error\_condition, 2116
  - std::regex\_traits< \_Ch\_type >, 2828
- value\_comp
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2404
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 2498
  - std::multiset< \_Key, \_Compare, \_Alloc >, 2522
  - std::set< \_Key, \_Compare, \_Alloc >, 2951
- value\_compare
  - std::set< \_Key, \_Compare, \_Alloc >, 2933
- value\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1724
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1729
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1739
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1742
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, 2306
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_const\_iterator< Node, \_Alloc >, 2310
  - std::allocator\_traits< \_Alloc >, 1034
  - std::allocator\_traits< allocator< \_Tp > >, 1040
  - std::allocator\_traits< allocator< void > >, 1049
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, 1057, 1058
  - std::back\_insert\_iterator< \_Container >, 1077
  - std::complex< \_Tp >, 1866
  - std::complex< double >, 1868
  - std::complex< float >, 1871
  - std::complex< long double >, 1874
  - std::front\_insert\_iterator< \_Container >, 2163
  - std::insert\_iterator< \_Container >, 2253
  - std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2292
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 2295
  - std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 2300
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 2687
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 2691
  - std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 2796
  - std::set< \_Key, \_Compare, \_Alloc >, 2933
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3172
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3202
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3231
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3257
- Variable template for type traits, 291
- variant, 3786
- vector, 3786–3788
  - std::\_\_debug::vector< \_Tp, \_Allocator >, 3288
  - std::vector< \_Tp, \_Alloc >, 3292–3294
  - std::vector< bool, \_Alloc >, 3312–3314
- vector.tcc, 3535
- version.h, 3535
- void\_pointer
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 735
  - std::allocator\_traits< \_Alloc >, 1034
  - std::allocator\_traits< allocator< \_Tp > >, 1040
  - std::allocator\_traits< allocator< void > >, 1049
  - std::allocator\_traits< pmr::polymorphic\_allocator< \_Tp > >, 1058
- void\_t
  - Detection idiom, 285
- vstring.h, 3674
- vstring.tcc, 3677
- vstring\_fwd.h, 3678
- vstringutil.h, 3679
- wbuffer\_convert
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3330, 3331
- wcerr
  - std, 646
- wcin
  - std, 646
- wclog
  - std, 647
- wcout
  - std, 647
- wcregex\_token\_iterator
  - Regular Expressions, 255
- wcsub\_match
  - Regular Expressions, 255
- weeks
  - Time, 366
- wfilebuf
  - I/O, 166

wfstream

I/O, [166](#)

what

\_\_gnu\_cxx::forced\_error, [2136](#)

\_\_gnu\_cxx::recursive\_init\_error, [2810](#)

\_\_gnu\_pbds::container\_error, [1897](#)

\_\_gnu\_pbds::insert\_error, [2251](#)

\_\_gnu\_pbds::join\_error, [2302](#)

\_\_gnu\_pbds::resize\_error, [2831](#)

std::bad\_alloc, [1079](#)

std::bad\_cast, [1081](#)

std::bad\_exception, [1082](#)

std::bad\_function\_call, [1083](#)

std::bad\_typeid, [1084](#)

std::bad\_weak\_ptr, [1085](#)

std::domain\_error, [2067](#)

std::exception, [2119](#)

std::experimental::filesystem::v1::filesystem\_error, [2130](#)

std::experimental::fundamentals\_v1::bad\_any\_cast, [1080](#)

std::experimental::fundamentals\_v1::bad\_optional\_access, I/O, [166](#)  
[1084](#)

std::filesystem::filesystem\_error, [2131](#)

std::future\_error, [2179](#)

std::invalid\_argument, [2256](#)

std::length\_error, [2312](#)

std::logic\_error, [2359](#)

std::out\_of\_range, [2693](#)

std::overflow\_error, [2702](#)

std::range\_error, [2782](#)

std::regex\_error, [2816](#)

std::runtime\_error, [2843](#)

std::system\_error, [3045](#)

std::underflow\_error, [3141](#)

widen

std::\_ctype\_abstract\_base<\_CharT>, [761](#)

std::basic\_fstream<\_CharT, \_Traits>, [1153](#)

std::basic\_ifstream<\_CharT, \_Traits>, [1196](#)

std::basic\_ios<\_CharT, \_Traits>, [1219](#)

std::basic\_iostream<\_CharT, \_Traits>, [1265](#)

std::basic\_istream<\_CharT, \_Traits>, [1303](#)

std::basic\_istream<\_CharT, \_Traits>::sentry, [2892](#)

std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1342](#)

std::basic\_ofstream<\_CharT, \_Traits>, [1376](#)

std::basic\_ostream<\_CharT, \_Traits>, [1406](#)

std::basic\_ostream<\_CharT, \_Traits>::sentry, [2919](#)

std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1440](#)

std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1710](#)

std::ctype<\_CharT>, [1922](#)

std::ctype<char>, [1955–1957](#)

std::ctype<wchar\_t>, [1975](#), [1976](#)

std::ctype\_byname<\_CharT>, [1989](#), [1990](#)

std::ctype\_byname<char>, [2011](#), [2012](#)

width

std::basic\_fstream<\_CharT, \_Traits>, [1154](#)

std::basic\_ifstream<\_CharT, \_Traits>, [1197](#)

std::basic\_ios<\_CharT, \_Traits>, [1220](#)

std::basic\_iostream<\_CharT, \_Traits>, [1266](#)

std::basic\_istream<\_CharT, \_Traits>, [1303](#), [1304](#)

std::basic\_istream<\_CharT, \_Traits>::sentry, [2892](#)

std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1343](#)

std::basic\_ofstream<\_CharT, \_Traits>, [1376](#)

std::basic\_ostream<\_CharT, \_Traits>, [1406](#), [1407](#)

std::basic\_ostream<\_CharT, \_Traits>::sentry, [2919](#)

std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1440](#)

std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1711](#)

std::ios\_base, [2265](#)

wfstream

wios  
I/O, [166](#)

wiostream  
I/O, [166](#)

wistream  
I/O, [166](#)

wistreamstream  
I/O, [166](#)

wofstream  
I/O, [166](#)

workstealing.h, [3747](#)

wostream  
I/O, [166](#)

wostringstream  
I/O, [166](#)

wregex  
Regular Expressions, [255](#)

write

std::basic\_fstream<\_CharT, \_Traits>, [1154](#)

std::basic\_iostream<\_CharT, \_Traits>, [1266](#)

std::basic\_ofstream<\_CharT, \_Traits>, [1377](#)

std::basic\_ostream<\_CharT, \_Traits>, [1407](#)

std::basic\_ostream<\_CharT, \_Traits>::sentry, [2920](#)

std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1441](#)

std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [1711](#)

ws

std, [645](#)

wsregex\_token\_iterator  
Regular Expressions, [255](#)

wssub\_match

- Regular Expressions, [255](#)
- wstreambuf
  - I/O, [167](#)
- wstreampos
  - std, [580](#)
- wstring
  - Strings, [273](#)
- wstring\_convert
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, [3348](#), [3349](#)
- wstringbuf
  - I/O, [167](#)
- wstringstream
  - I/O, [167](#)
- xalloc
  - std::basic\_fstream< \_CharT, \_Traits >, [1155](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1197](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1220](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1267](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1304](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2892](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1343](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1377](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1407](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2920](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1441](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [1712](#)
  - std::ios\_base, [2265](#)
- xsggetn
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2098](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3005](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3023](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1106](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1471](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1666](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3342](#)
- xspn
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2099](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3005](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3024](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1107](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1472](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [1667](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3342](#)
- years
  - Time, [366](#)
- yield
  - std::this\_thread, [721](#)